



# BEA WebLogic Adapter for CORBA™

## User Guide

# Copyright

Copyright © 2003 BEA Systems, Inc. All Rights Reserved.

Copyright © 2003 iWay Software. All Rights Reserved.

## Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

## Trademarks or Service Marks

BEA, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop and How Business Becomes E-Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

### BEA WebLogic Adapter for CORBA User Guide

| Part Number | Date       |
|-------------|------------|
| N/A         | April 2003 |

---

# Table of Contents

## About This Document

|                                 |      |
|---------------------------------|------|
| What You Need to Know .....     | vii  |
| Related Information.....        | viii |
| Contact Us!.....                | viii |
| Documentation Conventions ..... | ix   |

## 1. Introduction

|                         |     |
|-------------------------|-----|
| Overview .....          | 1-1 |
| Adapter Operation ..... | 1-2 |

## 2. Using Application Explorer With CORBA

|  |     |
|--|-----|
| Overview .....                               | 2-1 |
| Connecting to an Object Request Broker ..... | 2-2 |
| Creating Service Schemas .....               | 2-6 |

## 3. Creating an Application View for a CORBA Object

|   |      |
|---|------|
| Overview .....  | 3-1  |
| Creating an Application View Folder.....              | 3-2  |
| Creating an Application View for a CORBA Object ..... | 3-3  |
| Adding a CORBA Service to an Application View .....   | 3-7  |
| Testing an Application View .....                     | 3-14 |

## 4. Using Tracing

|                                       |     |
|---------------------------------------|-----|
| Levels and Categories of Tracing..... | 4-2 |
| Tracing and Performance .....         | 4-3 |
| Creating Traces for Services .....    | 4-3 |

---

## **A. Using CORBA Implementations With the Adapter**

|  |     |
|--|-----|
| Using JacORB .....                                   | A-1 |
| JacORB Name Service .....                            | A-2 |
| JacORB Interface Repository .....                    | A-2 |
| Building and Running the JacORB Request Broker ..... | A-3 |
| Using Orbix2000 .....                                | A-6 |
| Using VisiBroker for Java .....                      | A-7 |
| Compiling and Running the Club Package .....         | A-9 |

## **B. Supported IDL Types**

## **C. Sample Files**

|   |      |
|---|------|
| Sample XML Request and Response Documents ..... | C-1  |
| addGetReservation Request.....                  | C-3  |
| addGetReservation Response .....                | C-4  |
| AddReservation Request .....                    | C-4  |
| AddReservation Response.....                    | C-5  |
| addReservationComplex Request.....              | C-5  |
| addReservationComplex Response .....            | C-6  |
| addReservationWithOut Request .....             | C-8  |
| addReservationWithOut Response.....             | C-8  |
| cancelReservation Request.....                  | C-9  |
| cancelReservation Response .....                | C-9  |
| getClub Request.....                            | C-9  |
| GetClubNames Request.....                       | C-10 |
| GetClubNames Response .....                     | C-10 |
| getClubNames4 Request .....                     | C-10 |
| getClubNames4 Response .....                    | C-11 |
| getClubNamesWithRef Request.....                | C-11 |
| GetClubPrices Request.....                      | C-11 |
| GetClubPrices Response .....                    | C-12 |
| GetClubPricesAsArray Request .....              | C-13 |
| GetClubPricesAsArray Response.....              | C-13 |
| GetReservation Request .....                    | C-14 |
| GetReservation Response .....                   | C-14 |

---

|                                     |      |
|-------------------------------------|------|
| getReservationAsOut Request.....    | C-15 |
| getReservationAsOut Response .....  | C-15 |
| setCancel Request .....             | C-16 |
| setCancel Response .....            | C-16 |
| setClubPrices Request .....         | C-17 |
| setClubPrices Response.....         | C-17 |
| setClubPricesAsArray Request .....  | C-17 |
| setClubPricesAsArray Response.....  | C-18 |
| setStatus Request .....             | C-18 |
| setStatus Response .....            | C-19 |
| status Request .....                | C-19 |
| status Response .....               | C-19 |
| testOneDim Request.....             | C-20 |
| testOneDim Response .....           | C-20 |
| testTwoDim Request .....            | C-20 |
| testTwoDim Response.....            | C-21 |
| testTypes Request.....              | C-22 |
| testTypes Response .....            | C-22 |
| testUnion1-2 Request .....          | C-23 |
| testUnion2-1 Request .....          | C-23 |
| testUnion2-3 Request .....          | C-24 |
| testUnion3-PROCESSED Request .....  | C-24 |
| testUnion4-1 Request .....          | C-25 |
| testUnion4-default Request.....     | C-25 |
| Definitions for ClubMed Object..... | C-25 |
| JacORB.properties File .....        | C-26 |
| Build XML .....                     | C-35 |
| ClubMed.IDL File.....               | C-36 |
| ClubServer.java.jacorb File.....    | C-40 |



---

# About This Document

The *BEA WebLogic Adapter for CORBA User Guide* is organized as follows:

- [Chapter 1, “Introduction,”](#) introduces the BEA WebLogic Adapter for CORBA, describes its features, and presents an overview of how it works.
- [Chapter 2, “Using Application Explorer With CORBA,”](#) describes how to use the BEA Application Explorer to create service schemas.
- [Chapter 3, “Creating an Application View for a CORBA Object,”](#) describes how to create application views and add services to them.
- [Chapter 4, “Using Tracing,”](#) describes how to generate different kinds of trace logs to diagnose problems and provide audit trails.
- [Appendix A, “Using CORBA Implementations With the Adapter,”](#) provides details for Orbix2000, VisiBroker for Java, and JacORB.
- [Appendix B, “Supported IDL Types,”](#) lists the IDL variable types supported by the adapter.
- [Appendix C, “Sample Files,”](#) describe the sample files provided with the adapter.

## What You Need to Know

This document is written for system integrators who develop client interfaces between CORBA and other applications. It describes how to use the BEA WebLogic Adapter for CORBA and how to develop application environments with specific focus on message integration. It is assumed that readers know Web technologies and have a general understanding of Microsoft Windows and UNIX systems.

---

# Related Information

The following documents provide additional information for the associated software components:

- *BEA WebLogic Adapter for CORBA Installation and Configuration Guide*
- *BEA WebLogic Adapter for CORBA Release Notes*
- *BEA Application Explorer Installation and Configuration Guide*
- BEA WebLogic Server installation and user documentation, which is available at the following URL:

[http://edocs.bea.com/more\\_wls.html](http://edocs.bea.com/more_wls.html)

- BEA WebLogic Integration installation and user documentation, which is available at the following URL:

[http://edocs.bea.com/more\\_wli.html](http://edocs.bea.com/more_wli.html)

## Contact Us!

Your feedback on the BEA WebLogic Adapter for CORBA documentation is important to us. Send us e-mail at [docsupport@bea.com](mailto:docsupport@bea.com) if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the BEA WebLogic Adapter for CORBA documentation.

In your e-mail message, please indicate which version of the BEA WebLogic Adapter for CORBA documentation you are using.

If you have any questions about this version of the BEA WebLogic Adapter for CORBA, or if you have problems using the BEA WebLogic Adapter for CORBA, contact BEA Customer Support through BEA WebSupport at [www.bea.com](http://www.bea.com). You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:



- 
- Your name, e-mail address, phone number, and fax number
  - Your company name and company address
  - Your machine type and authorization codes
  - The name and version of the product you are using
  - A description of the problem and the content of pertinent error messages

## Documentation Conventions

The following documentation conventions are used throughout this document.

| Convention                             | Item  |
|--|---|
| <b>boldface text</b>                   | Indicates terms defined in the glossary.  |
| Ctrl+Tab                               | Indicates that you must press two or more keys simultaneously.  |
| <i>italics</i>                         | Indicates emphasis or book titles.  |
| monospace<br>text                      | Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.<br><br><i>Examples:</i><br><pre>#include &lt;iostream.h&gt; void main ( ) the pointer psz<br/>chmod u+w *<br/>\tux\data\ap<br/>.doc<br/>tux.doc<br/>BITMAP<br/>float</pre> |
| <b>monospace<br/>boldface<br/>text</b> | Identifies significant words in code.<br><i>Example:</i><br><pre>void <b>commit</b> ( )</pre>   |

---

| Convention                                       | Item  |
|--|---|
| <i>monospace</i><br><i>italic</i><br><i>text</i> | Identifies variables in code.<br><i>Example:</i><br>String <i>expr</i>  |
| UPPERCASE<br>TEXT                                | Indicates device names, environment variables, and logical operators.<br><i>Examples:</i><br>LPT1<br>SIGNON<br>OR   |
| { }  | Indicates a set of choices in a syntax line. The braces themselves should never be typed.   |
| [ ]  | Indicates optional items in a syntax line. The brackets themselves should never be typed.<br><i>Example:</i><br>buildobjclient [-v] [-o name ] [-f <i>file-list</i> ]...<br>[-l <i>file-list</i> ]...   |
|  | Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.   |
| ...  | Indicates one of the following in a command line: <ul style="list-style-type: none"><li>■ That an argument can be repeated several times in a command line</li><li>■ That the statement omits additional optional arguments</li><li>■ That you can enter additional parameters, values, or other information</li></ul> The ellipsis itself should never be typed.<br><i>Example:</i><br>buildobjclient [-v] [-o name ] [-f <i>file-list</i> ]...<br>[-l <i>file-list</i> ]... |
| .<br>. .<br>.                                    | Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.  |

# 1 Introduction

This section introduces the BEA WebLogic Adapter for CORBA, describes its features, and provides an overview of how it works. It includes the following topics:

- [Overview](#)
- [Adapter Operation](#)

## Overview

A number of companies and application providers have used Common Object Request Broker Architecture (CORBA) for internet and legacy C++ application development, particularly before the popularity of Java or J2EE and XML or Web Service architectures. The BEA WebLogic Adapter for CORBA integrates existing CORBA services with WebLogic Server and WebLogic Integration so that existing IT investments can be integrated into J2EE applications and deployed as Web services.

The BEA WebLogic Adapter for CORBA enables applications integrated by BEA WebLogic Integration to communicate with Common Object Request Broker Architecture (CORBA)-based applications. Access to CORBA environments is provided through the adapter, which uses CORBA Interface Definition Language (IDL) entries to generate local, remote-based services. Applications make calls to a remote service that, in turn, invokes a CORBA method and receives return information from the CORBA method.

A CORBA object provides distributed object capability between applications in a network. Although a CORBA object is implemented using a standard programming language, each CORBA object has a clearly defined interface, defined using the CORBA Interface Definition Language (IDL). The definition of a CORBA object is

consistent with the definition presented by the Object Management Group (OMG). OMG has a number of specifications and documents that provide complete details on objects. For more information, visit the OMG web site, located at <http://www.omg.org>.

The BEA WebLogic Adapter for CORBA provides a means to exchange real-time business data between CORBA servers and other application, database, or external business partner systems. The adapter allows for inbound and outbound processing with CORBA.

The adapter uses WebLogic Integration and XML messages to allow non-CORBA applications to communicate and exchange transactions with CORBA. Applications that need to cause a CORBA business event use the adapter in combination with WebLogic Integration application views, services, and business process workflows to send request messages to CORBA. If the request is for retrieving data from CORBA, then the adapter sends the application a response message with the data.

The BEA WebLogic Adapter for CORBA provides these key features:

- *Support for several ORBs*, such as JacORB, Orbix2000, and VisiBroker for Java.
- *Guaranteed synchronous and asynchronous message interactions* between WebLogic Integration and an object request broker (ORB).
- *Data transfer* between a business process running within WebLogic Integration and an ORB.
- *Service adapter integration operations* providing end-to-end business process management using XML schemas.
- *BEA Application Explorer*, which uses CORBA Object Manager meta data to build XML schemas for application view services.

## Adapter Operation

When the BEA WebLogic Adapter for CORBA receives a request for a service, the adapter converts the XML request document to a Dynamic Invocation Interface (DII) call and sends it to the CORBA server using the Internet Inter-ORB Protocol (IIOP). The response of the CORBA server is returned over IIOP. The adapter then formats the response as an XML response document for WebLogic Integration.

To create an application view from a CORBA server, you provide the adapter with configuration information for an Interface Repository (IFR) and either:

- An indirect object reference via a Naming Service location, ORB name context, and object name.
- A direct object reference via an Interoperable Object Reference.

The BEA Application Explorer displays a list of available systems. You can expand a system to display available servers and interfaces that were specified using CORBA IDL. You can expand an interface and create XSD service schemas, which are exported to WebLogic Integration as application views.

If communicating with the Naming Service, the adapter uses CORBA APIs to retrieve Interoperable Object References the first time they are needed. When the application view schema is created, BEA Application Explorer uses the adapter to retrieve the server's definition from the Interface Repository. The information is converted into XML schemas for use by WebLogic Integration.

If a server's interface changes, you must refresh the definitions that the adapter is using in order to avoid an out-of-date adapter run-time call, which would cause the call to fail if the new interface is incompatible with the old one. To refresh the definitions, stop the adapter processes from the WebLogic Integration console, and reload the new definitions the next time you view the system in the BEA Application Explorer.



# 2 Using Application Explorer With CORBA

This section describes how to connect to an Object Request Broker and create service schemas. It includes the following topics:

- [Overview](#)
- [Connecting to an Object Request Broker](#)
- [Creating Service Schemas](#)

## Overview

The BEA Application Explorer supports the creation of schemas based on specific tables and resulting answer sets. To obtain metadata about the Object Request Broker (ORB), the BEA Application Explorer connects to the Interface Repository. The BEA WebLogic Adapter for CORBA extracts the definition of CORBA servers and converts them to XML schemas and service XML request and response definitions. You can see the original definitions of the CORBA servers using the BEA Application Explorer.

The BEA Application Explorer displays a tree of all defined systems. By drilling down into the tree, you can determine which CORBA servers are available and how their interfaces will appear. Before you can view a CORBA server, you must ensure that the server is registered in the Naming Service and its IDL is loaded into the Interface Repository (IFR). After creating a connection in the BEA Application Explorer, you can use the Explorer to verify that the system definition was entered correctly.

Using the BEA WebLogic Adapter for CORBA, the BEA Application Explorer populates each system folder with object data retrieved from both the Naming Service and the Interface Repository. The contents of the Naming Service and Interface Repository appear in the Interfaces folder and the Objects folder, respectively.

The Objects folder contains all the servers registered in the Naming Service. If you expand a server you can see the interface that is implemented by that server. Expanding this interface displays its methods, return arguments, and parameters. Additional folders may appear under the Objects folder if additional modules were created. The Interfaces folder displays a list of modules, appearing as folders. These IFR modules represent the different work areas of CORBA.

The XML schema defines the format of XML requests and corresponding replies to the service adapter. The schema is a language-neutral interface description in XML format that declares the types, objects, and methods for the CORBA system. Conceptually, the XML schema is the same as the CORBA IDL.

**Note:** Before creating schemas and application view services in the BEA WebLogic Adapter for CORBA, you can save time by verifying that your ORB infrastructure is properly configured, your server is registered in the Naming Service, and your interface repository (IFR) is running and populated.

# Connecting to an Object Request Broker

To connect to an Object Request Broker (ORB):

1. Start the BEA Application Explorer:
  - *For JacORB*, from the command line run `ae.bat` (on a Windows system) or `ae` (on a UNIX system) under `installation_directory/bin`.
  - *For all other ORBs*, choose Start→Programs→BEA Application Explorer (on a Windows system), or run shell script `ae` under `installation_directory/bin` (on a UNIX system).

When you start the BEA Application Explorer, the left pane displays all the adapters supported by your version.

2. From the File menu, choose Session to change the default session path.



**Figure 2-1 Choosing a Session in BEA Application Explorer**

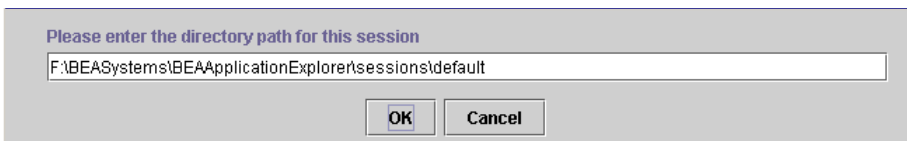
The Enter Session Path window opens, displaying the default session path.

The session path holds the schemas you that generate and your ORB connection information:

*session\_path\corba\connection\_name\schemas*

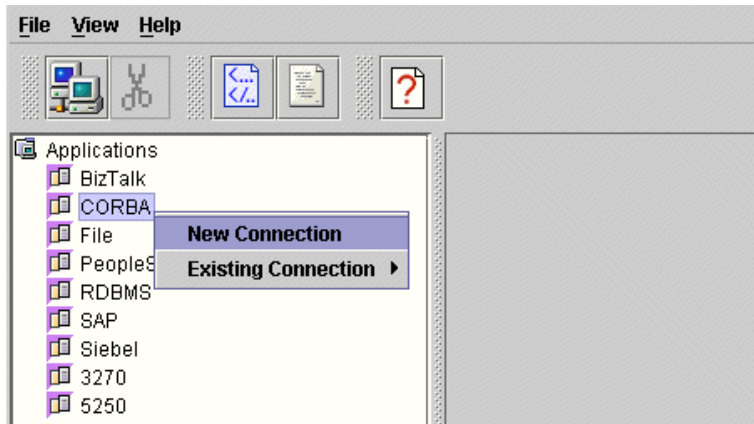
3. If you want to accept the default session path, click OK. Otherwise, to specify a different path, enter the path.

For example, you may want to specify a path for a particular project or for a logical grouping of services and events.

**Figure 2-2 Enter Session Path Dialog Box**

4. You can define a new connection to an ORB or use an existing connection:
  - To define a new connection to an ORB, right-click CORBA→New Connection. A dialog box opens prompting you for a connection name; continue with step 5.
  - To use an existing connection, right click CORBA→Existing Connection→*your connection*. The connection is displayed below the CORBA node in the left pane; skip ahead to step 7.

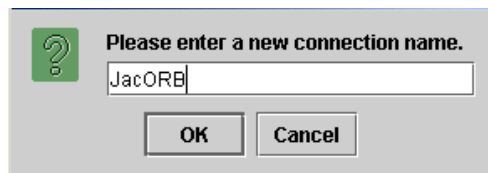
**Figure 2-3** Selecting a New Connection in BEA Application Explorer



5. Enter a connection name and click OK.

In the following figure, JacORB is the name of a logical connection.

**Figure 2-4** New Connection Name Input Window



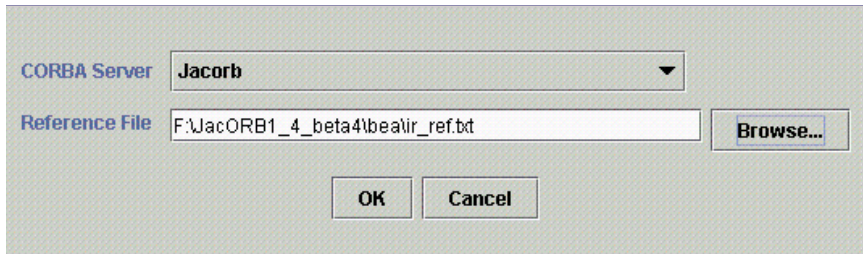
The ORB Logon window opens.

6. Select a CORBA Server and specify the name and path the Interface Repository reference file.

This file contains the parameters required to connect to the CORBA system.

**Note:** The example presented here illustrates accessing JacORB services. For other ORBs, there are additional parameters, such as Naming Service, Host, Port, and Enable IIOP Tracing. For details, see your ORB documentation.

**Figure 2-5 ORB Logon Window**

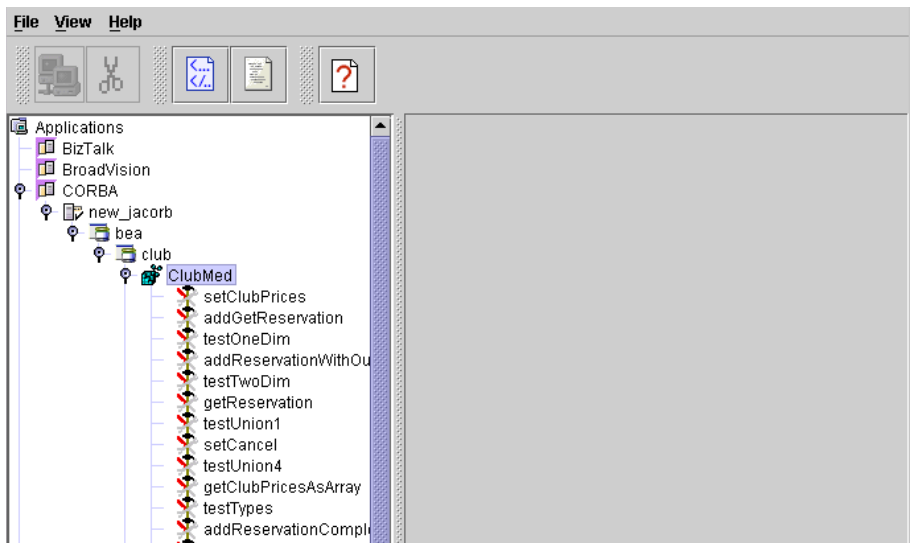


The BEA Application Explorer loads the application information and connects to the ORB to extract the object definitions from the Interface Repository.

7. You can now expand the tree to browse the available objects.

The following figure shows all available business services in a JacORB sample system called ClubMed.

**Figure 2-6 Available Objects Window**



For information about creating service schemas, see [“Creating Service Schemas”](#) on page 6.

# Creating Service Schemas

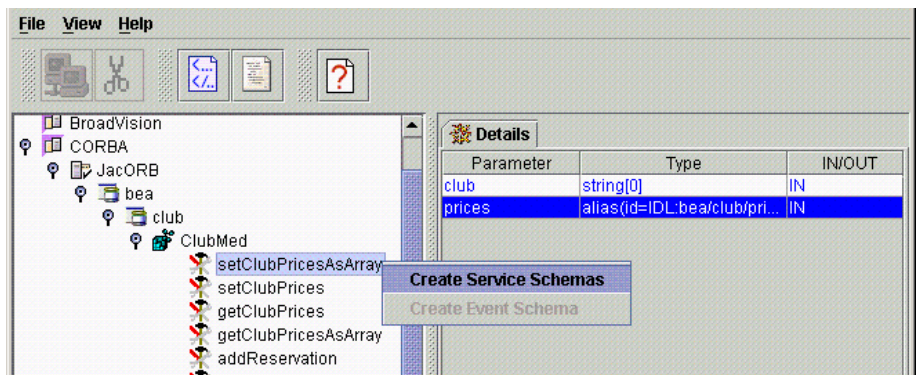
The BEA Application Explorer generates the following WebLogic Integration schemas:

- Service XML request schemas.
- Service XML response schemas.

To create service schemas:

1. Open the BEA Application Explorer and browse an object.
2. Right-click the desired service and choose Create Service Schemas.

**Figure 2-7 Service Schema Creation Window**

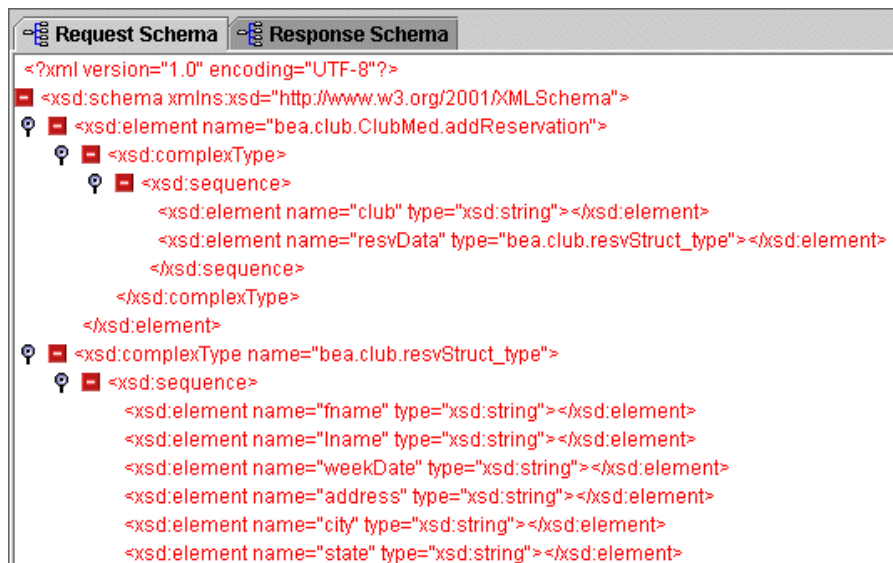


The BEA Application Explorer accesses the CORBA Interface Repository via IIOP and builds XSD schemas, which are then published to the WebLogic Integration Repository. You can view the request and response schemas for the CORBA interface object.

Request Schema and Response Schema tabs open in the right pane.

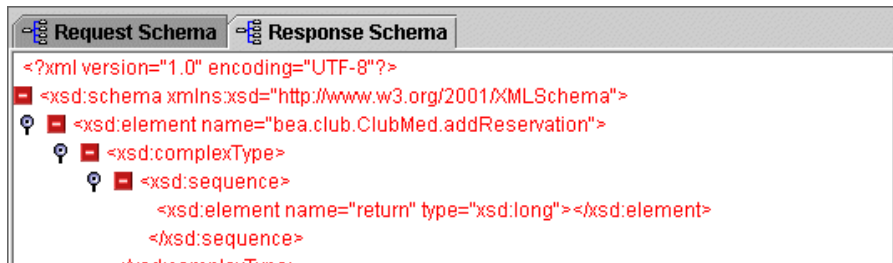
3. To view the request schema, click the Request Schema tab in the right pane.

**Figure 2-8 Request Schema Tab**



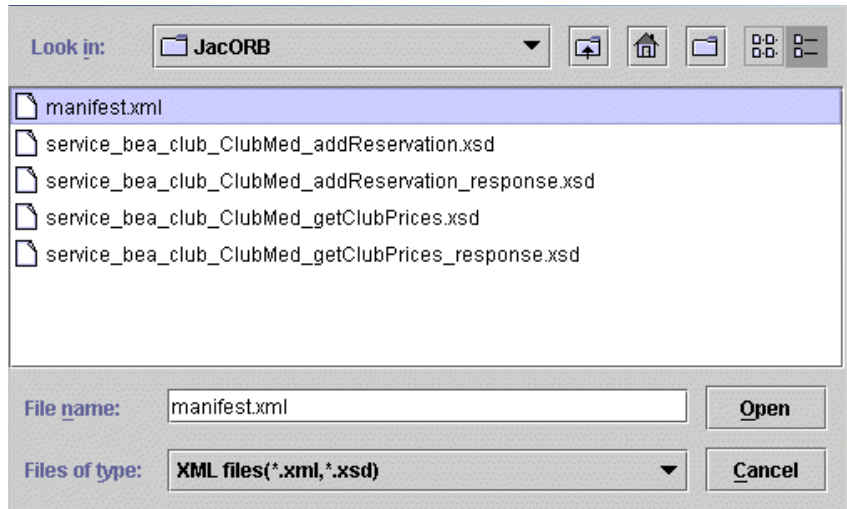
4. To view the response schema, click the Response Schema tab in the right pane.

**Figure 2-9 Response Schema Tab**



5. To view the manifest.xml file, choose View→View XML.

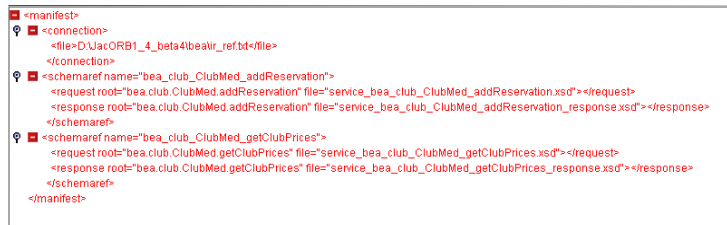
**Figure 2-10 View XML Window**



6. Click `manifest.xml`.

The `manifest.xml` file opens, as in the following figure:

**Figure 2-11 Sample manifest.xml Window**



The BEA WebLogic Adapter for CORBA uses `manifest.xml` and the service schemas to manage the interaction between an application view and the ORB.

# 3 Creating an Application View for a CORBA Object

This section describes how to create application views and add services to them. It includes the following topics:

- [Overview](#)
- [Creating an Application View Folder](#)
- [Creating an Application View for a CORBA Object](#)
- [Adding a CORBA Service to an Application View](#)
- [Testing an Application View](#)

## Overview

The BEA WebLogic Adapter for CORBA is a service adapter that is capable of processing IIOP CORBA object methods embedded in XML requests, and forwarding them to an Object Request Broker (ORB). The ORB object returns the data to the adapter, which in turn returns it to the client.

A service adapter performs the following functions in sequence:

1. Receives a service request from an external client.

2. Transforms the XML request document, which conforms to the services's request schema, into an ORB-specific format.
3. Invokes the underlying ORB function and waits for a response.
4. Transforms the response from the ORB-specific data format to an XML document that conforms to the service's response schema.

A service can be invoked asynchronously or synchronously. When a service is invoked asynchronously, the client application issues a service request and then proceeds with other processing. The client application does not wait for a response. When a service is invoked synchronously, the client waits for the response before proceeding with further processing. WebLogic Integration supports both these invocation methods, so you do not need to provide this functionality in your own application code.

## Creating an Application View Folder

Application views reside within WebLogic Integration. WebLogic Integration provides you with a root folder in which you can store all of your application views; if you wish, you can create additional folders to organize related application views into groups.

To create an application view folder:

1. Log on to the WebLogic Integration Application View Console at `//appserver-host:port/wlai`.

Here, *appserver-host* is the IP address or host name where the WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

2. If prompted, enter a user name and password.

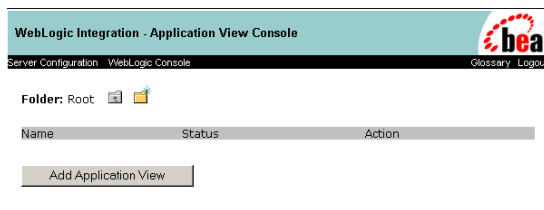
**Note:** If the user name is not `system`, it must be included in the adapter group. For more information on adding the administrative server user name to the adapter group, see the *BEA WebLogic Adapter for CORBA Installation and Configuration Guide*.

3. Click Login.

The WebLogic Integration Application View Console opens.



**Figure 3-1 Application View Console Main Window**



4. Double-click the new folder icon. The Add Folder window opens.

**Figure 3-2 Add Folder Window**



5. Supply a name for the folder, and then click Save.

You have finished creating the application view folder. To create an application view, see [“Creating an Application View for a CORBA Object” on page 3](#).

## Creating an Application View for a CORBA Object

When you define an application view, you are creating an XML-based interface between WebLogic Server and a particular CORBA Object Request Broker (ORB) within your enterprise. Once you create the application view, a business analyst can create business processes that use the application view. You can create any number of application views, each with any number of services. For more information, see:

- “Defining an Application View” in *Using Application Integration*
- <http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

### 3 Creating an Application View for a CORBA Object

---

To create an application view:

1. Log on to the WebLogic Application View Console at  
`//appserver-host:port/wlai.`

Here, *appserver-host* is the IP address or host name where the WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

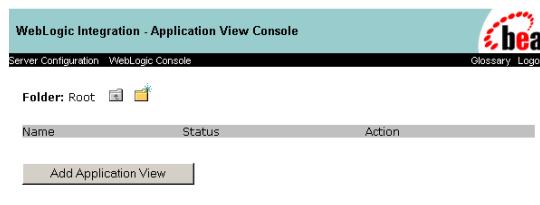
2. If prompted, enter a user name and password.

**Note:** If the user name is not *system*, it must be included in the adapter group. For more information on adding the administrative server user name to the adapter group, see the *BEA WebLogic Adapter for CORBA Installation and Configuration Guide*.

3. Click Login.

The WebLogic Integration Application View Console opens.

**Figure 3-3 Application View Console Window**



4. Click Add Application View. The Define New Application View window opens.

**Figure 3-4 Define New Application View Window**

**Define New Application View**

This page allows you to define a new application view

Folder: [Root](#)

Application View Name:\*

Description:

Associated Adapter:

5. In the Application View Name field, enter a name.

The name should describe the set of functions performed by this application. Each application view name must be unique to its adapter. Valid characters include a-z, A-Z, 0-9, and the \_ (underscore) character.

6. In the Description field, enter any relevant notes.

These notes are viewed by users when they use this application view in workflows.

7. From the Associated Adapter drop-down list, select `BEA_CORBA_1_0.ear` (the adapter ear file).

8. Click OK. The Configure Connection Parameters window opens.

**Table 3-1 Configure Connection Parameters Window**

**Configure Connection Parameters**

On this page, you supply parameters to connect to your EIS

The BEA Application Explorer generates schema information for a session stored at a location that must be known to the general adapter. Enter this session location here. A session can support multiple connections.

Once you have entered the **session path** location, click on the pulldown arrow for the **connection name**, which will display a selection list of valid connections.

Session Path\*

Connection Name\*

### 3 Creating an Application View for a CORBA Object

9. Enter the name of the BEA WebLogic Adapter for CORBA session path (sometimes known as the session base directory).

This is the path that you specify when starting a BEA Application Explorer session, as described in “[Connecting to an Object Request Broker](#)” in [Chapter 2](#), “[Using Application Explorer With CORBA.](#)” The session path holds your ORB schema and connection information.

10. Select the session name—also known as the connection name—from the Connection name drop-down list.
11. Click Connect to EIS. The Application View Administration window opens.

**Figure 3-5 Application View Administration Window**

Application View Administration for CORBA

Application View Console WebLogic Console

Configure Connection  
Administration  
Add Service  
Add Event  
Deploy Application View

This page allows you to add events and/or services to an application view.

Description: CORBA clubmed.idl [Edit](#)

Connection Criteria

Connection Name: JacORB

Additional Log Category: CORBA

Root Log Category: BEA\_CORBA\_1\_0

Session Path: D:\beaschemas

Message Bundle Base: BEA\_CORBA\_1\_0

Log Configuration File: BEA\_CORBA\_1\_0.xml

[Reconfigure connection parameters for CORBA](#)

Events [Add](#)

Services [Add](#)

[Save](#) ?

12. Click Save.

You have finished creating the application view.

You can now add services that support the application’s functions, as described in “[Adding a CORBA Service to an Application View](#)” on [page 3-7](#). Note that you must add a service before you deploy the new application view.

# Adding a CORBA Service to an Application View

To add a service to an application view:

1. If it is not already open, open the application view to be modified:

- a. Log on to the WebLogic Application View Console at

`//appserver-host:port/wlai`

Here, *appserver-host* is the IP address or host name where the WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

- b. If prompted, enter a user name and password.

**Note:** If the user name is not `system`, it must be included in the `adapter` group. For more information on adding the administrative server user name to the `adapter` group, see the *BEA WebLogic Adapter for CORBA Installation and Configuration Guide*.

- c. Click Login.

The WebLogic Integration Application View Console opens.

- d. Select the folder in which this application view resides, and then select the application view.

The Administration window opens.

### 3 Creating an Application View for a CORBA Object

**Figure 3-6 Application View Administration Window**

Application View Administration for CORBA

Application View Console   WebLogic Console

Configure Connection  
Administration  
Add Service  
Add Event  
Deploy Application View

This page allows you to add events and/or services to an application view.

Description: CORBA clubmed.idl [Edit](#)

Connection Criteria

|                          |                   |
|--------------------------|-------------------|
| Connection Name:         | JacORB            |
| Additional Log Category: | CORBA             |
| Root Log Category:       | BEA_CORBA_1_0     |
| Session Path:            | D:\beaschemas     |
| Message Bundle Base:     | BEA_CORBA_1_0     |
| Log Configuration File:  | BEA_CORBA_1_0.xml |

[Reconfigure connection parameters for CORBA](#)

Events

Services

Save ?

2. If the application view is deployed, you must undeploy it before adding the service. See “Optional Step: Undeploying an Application View” in “Defining an Application View” in *Using Application Integration*.

For WebLogic Integration 7.0, see

<http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

3. Click Add Service in the left pane.

The Add Service window opens.

**Figure 3-7 Add Service Window**

**Edit Service**

Application View Console WebLogic Console Glossary Logout

On this page, you edit service properties.

Unique Service Name: \*

Select:

|  |                                 |
|--|---------------------------------|
| Vendor Specific ORB Class*                 | org.jacorb.orb.ORB              |
| Vendor Specific ORB Singleton Class*       | org.jacorb.orb.ORBSingleton     |
| Interface_Repository_reference_(IOR)_file* | d:\apps\JacORB_1_4_1\VR_Ref.ior |
| Naming_Service_reference_(IOR)_file        | D:\apps\JacORB_1_4_1\NS_Ref.ior |
| Orb/Name Context                           | BEA                             |
| Object Name                                | ClubMed                         |
| Object_reference_(IOR)_file                |                                 |
| Timeout(ms)                                | 60000                           |

schema:

**settings**

|                       |                                     |
|-----------------------|-------------------------------------|
| Trace on/off          | <input checked="" type="checkbox"/> |
| Verbose Trace on/off  | <input checked="" type="checkbox"/> |
| Document Trace on/off | <input checked="" type="checkbox"/> |

- Specify the service's properties, which are described in the following table. Required values are indicated with an asterisk (\*).

**Table 3-2 Service Properties**

| Property                  | Description  |
|---------------------------|--|
| Unique Service Name       | The name of the service that you are adding.<br>The name must be unique within the application view. Valid characters include a-z, A-Z, 0-9, and underscore ( _ ).   |
| Select                    | The type of client ORB libraries through which the service will be sent.<br><br>The Application View Console automatically populates many of this window's fields with appropriate values based on the ORB type that you choose. |
| Vendor Specific ORB Class | The vendor-specific ORB class.<br>The Application View Console automatically supplies this value based on the ORB that you chose.  |

### 3 *Creating an Application View for a CORBA Object*

---

**Table 3-2 Service Properties (Continued)**

| Property                                  | Description   |
|---|---|
| Vendor Specific ORB Singleton Class       | <p>The vendor-specific ORB singleton class.</p> <p>The Application View Console automatically supplies this value based on the ORB that you chose.</p>  |
| Interface_Repository_reference_(IOR)_file | <p>The name and path of the Interoperable Object Reference file that specifies the location of the Interface Repository service.</p> <p>The Application View Console automatically supplies this value based on the value supplied in the BEA Application Explorer.</p>                   |
| Naming_Service_reference_(IOR)_file       | <p>The name and path of the Interoperable Object Reference file that specifies the location of the Naming Service.</p> <p>This parameter, together with Orb/Name Context and Object Name, enable you to specify the location of an object using an indirect Naming Service reference.</p> |
| Orb/Name Context                          | <p>The ORB's name context.</p> <p>This parameter, together with Orb/Name Context and Object Name, enable you to specify the location of an object using an indirect Naming Service reference.</p>   |
| Object Name                               | <p>The name of the object registered in the Naming Service (for example, <code>bea.clubmed</code>).</p> <p>This parameter, together with Orb/Name Context and Object Name, enable you to specify the location of an object using an indirect Naming Service reference.</p>                |
| Object_reference_(IOR)_file               | <p>The name and path of the Interoperable Object Reference file that specifies the location of the CORBA object.</p> <p>This parameter enables you to specify the location of an object using a direct IOR reference.</p>   |
| Timeout(ms)                               | <p>The maximum time that a service will wait for a CORBA object to respond before the service terminates. It is measured in milliseconds.</p> <p>The default value, 0, specifies that the service will wait indefinitely for a response.</p>  |
| Schema                                    | <p>The name of the schema that describes this service.</p>  |



**Table 3-2 Service Properties (Continued)**

| Property                             | Description  |
|--------------------------------------|--|
| Trace on/off                         | Generates a basic trace that displays the input XML (up to 300 bytes) before parsing, and shows the request being processed. For more information about tracing, see <a href="#">Chapter 4, “Using Tracing.”</a> |
| Verbose Trace on/off                 | Generates a trace that displays configuration parameters used by the adapter. For more information about tracing, see <a href="#">Chapter 4, “Using Tracing.”</a>  |
| Document Trace on/off                | Generates a trace that displays the input document after it was analyzed and the response document being returned. For more information about tracing, see <a href="#">Chapter 4, “Using Tracing.”</a>           |
| root_to_transform_template_directory | The directory from which an .xch transform template file can be loaded for the XCH transform type in the XML to XML transformation phase.  |
| root_to_XML_Style_sheet_directory    | The directory from which an XSLT style sheet can be loaded for the XSLT transform type in the XML to XML transformation phase.   |

5. Click Add.

The Administration window opens.

6. Click Continue.

The Deploy Application View window opens.

### 3 *Creating an Application View for a CORBA Object*

**Figure 3-8 Deploy Application View Window**

Deploy Application View CORBA to Server

Application View Console WebLogic Console

On this page you deploy your application view to the application server.

**Required Service Parameters**

Enable asynchronous service invocation? ☒

**Connection Pool Parameters**

Use these parameters to configure the connection pool used by this application view

Minimum Pool Size\* 1

Maximum Pool Size\* 10

Target Fraction of Maximum Pool Size\* 0.7

Allow Pool to Shrink? ☒

**Log Configuration**

Set the log verbosity level for this application view.

Log warnings, errors, and audit messages

**Configure Security**

[Restrict Access to CORBA using J2EE Security](#)

Deploy ☒ Deploy persistently? ☒ Save

7. Update service parameters, connection pool parameters, log configuration, and security as necessary. For more information about these, see “Defining an Application View” in “Using Application Integration”.

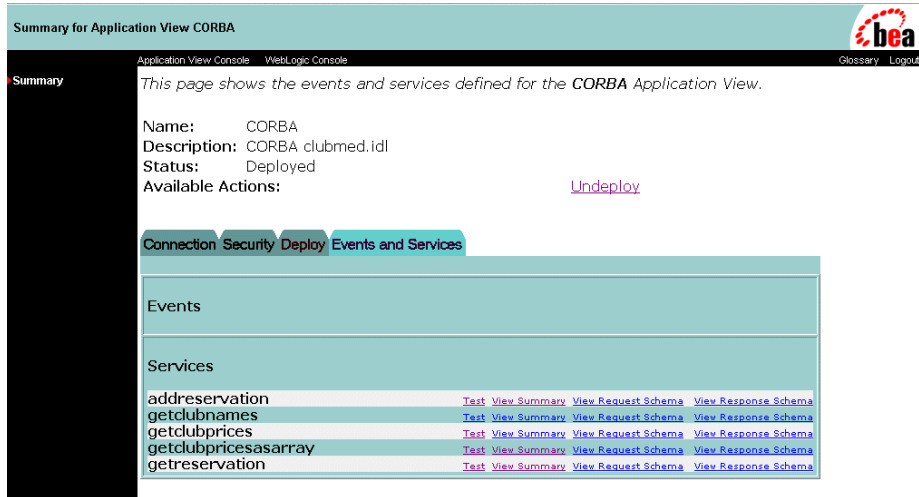
For WebLogic Integration 7.0, see

<http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

8. Click Deploy to save and deploy the service.

The Summary window for the application view opens.

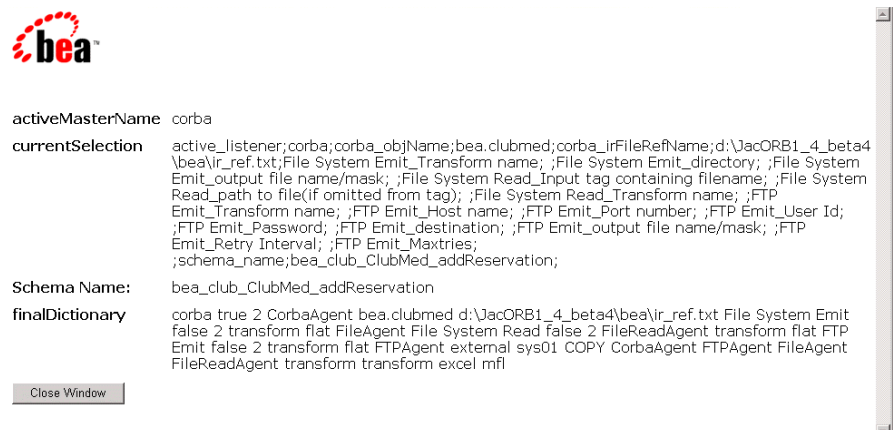
**Figure 3-9 Summary for Application View Window**



- To view a summary of the service as deployed, select the service and click View Summary.

A summary of the service is displayed in a new window.

**Figure 3-10 Service Summary Window**



You are now ready to test your application view, as described in “Testing an Application View” on page 14.

## Testing an Application View

To test that an application view service interacts properly with the BEA WebLogic Adapter for CORBA:

1. If it is not already open, open the application view to be tested:

- a. Log on to the WebLogic Application View Console at

`//appserver-host:port/wlai`

Here, *appserver-host* is the IP address or host name where the WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

- b. If prompted, enter a user name and password.

**Note:** If the user name is not `system`, it must be included in the `adapter` group. For more information on adding the administrative server user name to the `adapter` group, see the *BEA WebLogic Adapter for CORBA Installation and Configuration Guide*.

- c. Click Login.

The WebLogic Integration Application View Console opens.

- d. Select the folder in which this application view resides, and then select the application view.

The Summary window opens.

**Figure 3-11 Displaying Services to Be Tested**

**Summary for Application View CORBA**

Application View Console WebLogic Console

**Summary**

This page shows the events and services defined for the **CORBA** Application View.

Name: CORBA  
 Description: CORBA clubmed.idl  
 Status: Deployed  
 Available Actions: [Undeploy](#)

**Connection Security Deploy Events and Services**

**Events**

**Services**

|                      |                      |                              |                                     |                                      |
|----------------------|----------------------|------------------------------|-------------------------------------|--------------------------------------|
| addressreservation   | <a href="#">Test</a> | <a href="#">View Summary</a> | <a href="#">View Request Schema</a> | <a href="#">View Response Schema</a> |
| getclubnames         | <a href="#">Test</a> | <a href="#">View Summary</a> | <a href="#">View Request Schema</a> | <a href="#">View Response Schema</a> |
| getclubprices        | <a href="#">Test</a> | <a href="#">View Summary</a> | <a href="#">View Request Schema</a> | <a href="#">View Response Schema</a> |
| getclubpricesasarray | <a href="#">Test</a> | <a href="#">View Summary</a> | <a href="#">View Request Schema</a> | <a href="#">View Response Schema</a> |
| getreservation       | <a href="#">Test</a> | <a href="#">View Summary</a> | <a href="#">View Request Schema</a> | <a href="#">View Response Schema</a> |

2. Select the service you want to test and click Test.

The Test Service window opens.

3. Enter a sample XML request document into the Test Service window, either by typing it or by copying and pasting it into the window. You can create your own request, or use one of the sample requests provided with the adapter and shown in [Appendix C, "Sample Files."](#)

The document should conform to the request schema of the service you are testing.

### 3 Creating an Application View for a CORBA Object

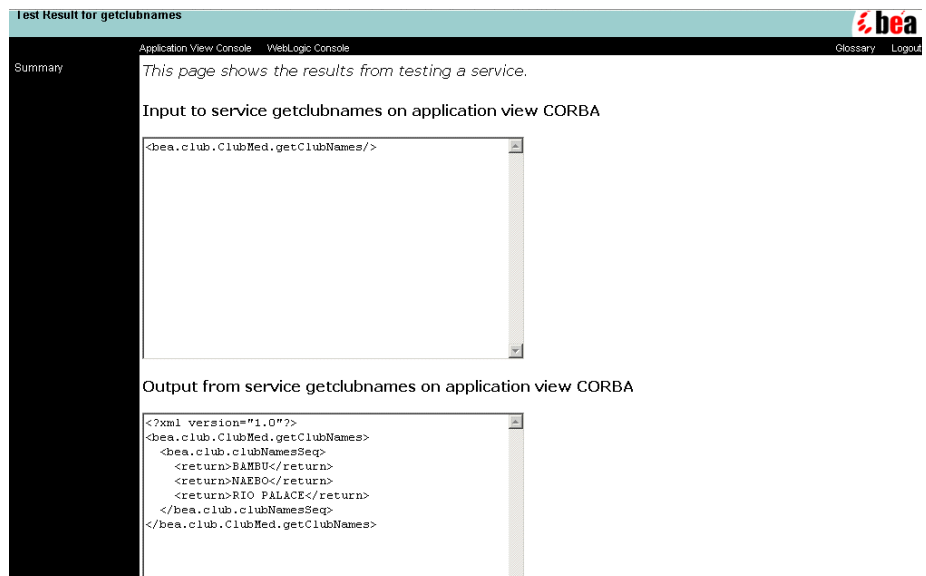
**Figure 3-12 Entering a Request Into the Test Service Window**



4. Click Test.

If the test is successful, the Test Result window displays both the request and the response. The application view is successfully deployed.

**Figure 3-13 Test Result Window**



# 4 Using Tracing

Tracing is an essential feature of an adapter. Most adapters integrate different applications and do not interact with end users while processing data. Unlike a front-end component, when an adapter encounters an error or a warning condition, the adapter cannot stop processing and wait for an end user to respond.

Many business applications that use adapters are mission-critical. Adapter components must provide both accurate logging and auditing information. For example, an adapter might maintain an audit report of every transaction with an Enterprise Information System (EIS). The adapter tracing and logging framework accommodates both logging and auditing.

The following sections describe tracing for services:

- [Levels and Categories of Tracing](#)
- [Tracing and Performance](#)
- [Creating Traces for Services](#)

# Levels and Categories of Tracing

Tracing is provided by both the BEA adapter framework and by the BEA WebLogic Adapter for CORBA. The BEA WebLogic Integration framework provides five distinct levels of tracing.

**Table 4-1 Tracing Levels**

| Level . . . | Indicates . . .   |
|-------------|---|
| AUDIT       | An extremely important log message related to the business processing performed by an adapter.<br>Messages with this priority always appear in the log. |
| ERROR       | An error in the adapter.<br>Adapters internationalize and localize error messages.  |
| WARN        | A situation that is not an error, but that might cause problems in the adapter.<br>Adapters internationalize and localize warn messages.                |
| INFO        | An informational message.<br>Adapters internationalize and localize info messages.  |
| DEBUG       | An informational message for debugging or troubleshooting.<br>Adapters do not usually internationalize and localize error messages.                     |

The adapter framework provides three specialized categories of tracing.

**Table 4-2 Specialized Tracing Categories**

| Level . . . | Indicates . . .   |
|-------------|---|
| Basic Trace | Basic traces.<br>The adapter displays the input XML (up to 300 bytes) before parsing, and shows the relevant request. The default setting is off. |



**Table 4-2 Specialized Tracing Categories**

| Level . . .    | Indicates . . .  |
|----------------|--|
| Verbose Trace  | More extensive traces.<br>The adapter displays configuration parameters. The default setting is off.   |
| Document Trace | Traces of the input document.<br>The adapter displays the input document after analyzing it and returning a response document. Because some documents are very large, this trace category can severely affect performance. The default setting is off. |

**Note:** You must declare both the level and category to obtain an appropriate trace. BEA Customer Support requests (minimally) a Basic and a Verbose trace when helping you debug.

## Tracing and Performance

The adapter provides additional trace capabilities that are not strictly hierarchic, but categorized. These traces provide debugging help with a minimum effect on performance. You may control all internal adapter traces through these additional tracing settings. The additional settings route their output to the standard debug setting.

If you configure the adapter for additional settings and do not configure standard trace settings, the traces are generated but never appear in output. This affects performance, as the production of the trace continues even though you receive no benefit of the additional trace information.

## Creating Traces for Services

To create traces for a service to diagnose adapter problems:

1. Create or modify the service.
2. Ensure all the adapter parameters are correct.

**Figure 4-1 Add Service Window**

**Edit Service**

Application View Console WebLogic Console

Glossary Logout

Configure Connection  
Administration  
Add Service  
Add Event  
Deploy Application View

On this page, you edit service properties.

Unique Service Name:\* getClubNames

Select: jacORBService

|  |                                 |
|--|---------------------------------|
| Vendor Specific ORB Class*                 | org.jacorb.orb.ORB              |
| Vendor Specific ORB Singleton Class*       | org.jacorb.orb.ORBSingleton     |
| Interface_Repository_reference_(IOR)_file* | d:\apps\JacORB_1_4_1\IR_Ref.ior |
| Naming_Service_reference_(IOR)_file        | D:\apps\JacORB_1_4_1\NS_Ref.ior |
| Orb/Name Context                           | BEA                             |
| Object Name                                | ClubMed                         |
| Object_reference_(IOR)_file                |                                 |
| Timeout(ms)                                | 60000                           |

schema: bea\_club\_ClubMed\_getClubNames

**settings**

|                       |                                     |
|-----------------------|-------------------------------------|
| Trace on/off          | <input checked="" type="checkbox"/> |
| Verbose Trace on/off  | <input checked="" type="checkbox"/> |
| Document Trace on/off | <input checked="" type="checkbox"/> |

3. Select the appropriate schema from the drop-down list.
4. Select the appropriate trace levels. For a list of trace levels, see the table titled “Tracing Levels” in the section “Levels and Categories of Tracing” earlier in this chapter.
5. Click Add to continue to the next configuration pane.
6. Click Continue to move to the next configuration pane.

The Deploy Application View window opens.

Figure 4-2 Deploy Application View Window

**Deploy Application View CORBA to Server**

Application View Console WebLogic Console

On this page you deploy your application view to the application server.

**Required Service Parameters**

Enable asynchronous service invocation? ☒

**Connection Pool Parameters**

Use these parameters to configure the connection pool used by this application view

Minimum Pool Size\*

Maximum Pool Size\*

Target Fraction of Maximum Pool Size\*

Allow Pool to Shrink? ☒

**Log Configuration**

Set the log verbosity level for this application view.

**Configure Security**

[Restrict Access to CORBA using J2EE Security](#)

☒ Deploy persistently?

7. Navigate to the Log Configuration area and select the desired trace level.

This pane enables you to select the trace level for the BEA WebLogic Integration framework.

For maximum tracing, select Log all Messages. This is recommended to obtain optimum debugging information for BEA support personnel.

**Note:** This causes all generated messages to be written to the log. You must select the desired category as defined in Specialized Tracing Categories in the adapter to generate the required messages.

8. Click Deploy (or Save) to set the trace settings and deploy the application view.

Traces are created the next time the service is invoked.

Traces are output to a file named BEA\_CORBA\_1\_0.log in the WebLogic Domain home directory.



# A Using CORBA Implementations With the Adapter

This section provides details about using the BEA WebLogic Adapter for CORBA with JacORB, Orbix2000, and VisiBroker for Java. It includes the following topics:

- [Using JacORB](#)
- [Using Orbix2000](#)
- [Using VisiBroker for Java](#)

## Using JacORB

The BEA WebLogic Adapter for CORBA includes a sample ORB called JacORB. JacORB is an open source Java implementation of the Object Management Group's CORBA specification. It is supplied with your adapter software to enable you to test the adapter.

JacORB is designed to comply with CORBA 2.3 Java language mapping, and supports commonly used CORBA services. It runs on all platforms that implement the Java Virtual Machine (JVM). JacORB is made available under the terms of the GNU Library General Public License (LGPL). Commercial support is provided by Object Computing Inc., a Sun Authorized Java Center and member of the OMG; for more information, go to <http://www.ociweb.com>.

JacORB operates with any CORBA-compliant ORB over IIOP. In practice, JacORB has been used successfully with at least the following ORBs: MICO, TAO, Orbacus, Iona Orbix, Borland VisiBroker, ORBit, omniORB, Vitria C++, and Java. ORB interoperability is made simple by using a foreign name service IOR in the file where the BEA WebLogic Adapter for CORBA looks up the name server OAR. This can be configured in the `jacorb.properties` file.

This section presents the following information about using JacORB with the Weblogic Adapter for CORBA:

- [JacORB Name Service](#)
- [JacORB Interface Repository](#)
- [Building and Running the JacORB Request Broker](#)

## **JacORB Name Service**

Name servers are used to locate objects using a human-readable reference (a name) rather than a machine or network address. If objects providing a certain service are looked up using the service name, clients are separated from the actual locations of the objects that provide the service. The binding from name to service can be changed without the client's knowledge.

JacORB provides an implementation of the OMG's Interoperable Naming Service (INS), which supports the binding of names to object references (and looking up object references using these names). It also allows clients to easily convert names to strings and vice versa. The JacORB name service comprises two components: the name server program, and a set of interfaces and classes used to access the service.

## **JacORB Interface Repository**

Run-time type information in CORBA is managed by the ORB's Interface Repository (IFR) component. It allows applications to request, inspect, and modify IDL type information dynamically. For example, the IFR enables applications to find out which operations an object supports. Some ORBs may also need the IFR to find out whether

a given object's type is a subtype of another, but most ORBs can operate without the IFR by encoding this kind of type information in the helper classes generated by the IDL compiler.

In essence, the IFR is just another remotely accessible CORBA object that offers operations to retrieve (and, theoretically, to modify) type information. The IFR manages type information in a hierarchical containment structure that corresponds to constructs within IDL specifications: modules contain definitions of interfaces, structures, constants, and so on. Interfaces in turn contain definitions of exceptions, operations, attributes, and constants.

## Building and Running the JacORB Request Broker

To build and run the JacORB Request Broker, perform the following steps. (You will need ANT 1.4.1, a Java-based build tool.)

1. Unzip `JacORB1_4_1-full.zip`.

This file is found in `BEA_CORBA_SAMPLES.zip` in the adapter's installation directory. It creates the directory `JacORB1_4_1` on the selected drive.

For example, if you unzip to drive D, the result is `D:\JacORB1_4_1`.

2. Unzip `beacorba.zip` in the directory `JacORB1_4_1` that you created in Step 1.

This file is found in `BEA_CORBA_SAMPLES.zip` in the adapter's installation directory.

3. Copy `JacORB1_4_1\jacorb_properties.template` and rename the copy `JacORB1_4_1\jacorb.properties`.

4. Edit the `jacorb.properties` file:

- a. In the file's Initial References Configuration section, uncomment this line:

```
#ORBInitRef.NameService=file:/d:/JacORB1_4_1/bea/ns_ref.ior
```

- b. In the same line, replace `d:` with the drive and path into which you unzipped your JacORB files.

- c. If the next line—`#ORBInitRef.NameService=file...`—is uncommented, comment it out.

5. Copy `JacORB1_4_1\bea\jaco.bat` into the `JacORB\bin` directory, replacing `JacORB1_4_1\bin\jaco.bat`.
6. Edit `JacORB1_4_1\bea\setenv-sample.bat` to specify directories in the following statements:

```
set JAVA_HOME=jdk_directory
set JACORB_HOME=JacORB1_4_1_directory
set ANT_HOME=ant_tool_directory
set PATH=%JACORB_HOME%\bin;%ANT_HOME%\bin;
%JAVA_HOME%\bin;%PATH%
set classpath=%JACORB_HOME%;%JACORB_HOME%\jacorb.properties;
%JACORB_HOME%\lib\jacorb.jar;%JACORB_HOME%\classes;
%ANT_HOME%\lib\ant.jar
```

Here, *jdk\_directory* is the directory where your JDK resides, *JacORB1\_4\_1\_directory* is the directory where JacORB resides, and *ant\_tool\_directory* is the directory where your ANT tool resides.

For example:

```
set JAVA_HOME=d:\apps\jdk1.3.1_04
set JACORB_HOME=d:\apps\JacORB1_4_1
set ANT_HOME=d:\apps\ant-1.5.2
set PATH=%JACORB_HOME%\bin;%ANT_HOME%\bin;
%JAVA_HOME%\bin;%PATH%
set classpath=%JACORB_HOME%;%JACORB_HOME%\jacorb.properties;
%JACORB_HOME%\lib\jacorb.jar;%JACORB_HOME%\classes;
%ANT_HOME%\lib\ant.jar
```

7. Rename `JacORB1_4_1\bea\club\ClubServer.java.jackorb` to `JacORB1_4_1\bea\club\ClubServer.java`
8. Build the JacORB application. In a new DOS command window:

- a. Execute `d:JacORB1_4_1\bea\setenv-sample.bat`
- b. Execute `d:JacORB1_4_1\bea\club\ant`

Here, *d:* is the drive and path into which you unzipped your JacORB files.

9. Start the Interface Repository service. In a new DOS command window:
  - a. Execute `JacORB1_4_1\bea\setenv-sample.bat`
  - b. Execute the following command:

```
ir repository_class_path IOR_filename
```



Here, *repository\_class\_path* is the path to your repository class files, and *IOR\_filename* is the name of the Interface Object Repository file.

For example:

```
JacORB1_4_1\bea\ir ..\classes ir_ref.txt
```

10. Start the name service. In a new DOS command window:

- a. Execute *d:\JacORB1\_4\_1\bea\setenv-sample.bat*

Here, *d:* is the drive and path into which you unzipped your JacORB files.

- b. Execute the following command:

```
ns [ins_filename] [-p port] [-t timeout]
```

Here, *ins\_filename* is the name of the Naming Service file specified in the *jacorb.properties* file, *port* is the number of the port on which the service is listening, and *timeout* is the server timeout.

For example:

```
JacORB1_4_1\bea\ns ns_ref.ior
```

11. Start JacORB's NameManager using the *nmg* command.

12. Start the Java interpreter explicitly by typing

- a. Execute *d:\JacORB1\_4\_1\bea\setenv-sample.bat*

Here, *d:* is the drive and path into which you unzipped your JacORB files.

- b. Execute the following command:

```
jaco jacobnaming.NameServer [filename] [-p port] [-t timeout]
```

Here, *jacobnaming.NameServer* is the name of the Name Server, *filename* is the name of the Naming Service file specified in the *jacorb.properties* file, *port* is the number of the port on which the service is listening, and *timeout* is the server timeout.

For example:

```
JacORB1_4_1\bea\jaco bea.club.ClubServer
```

## Using Orbix2000

The BEA WebLogic Adapter for CORBA supports Orbix 2000 Versions 1.2 and communicates using IIOP version 1.1.

Orbix is a software environment for building and integrating distributed object-oriented applications. Orbix is a full implementation of the Common Object Request Broker Architecture (CORBA) from the Object Management Group (OMG). Orbix fully supports CORBA version 2.3. For more information, see the *Orbix 2000 Administrator's Guide*.

Orbix includes a CORBA IDL compiler, which is used by programmers to compile interface definitions along with the client and server code. A client application compiled in this way contains internal information about server objects. Clients use this information to invoke the remote objects. Orbix provides an interface repository, which enables clients to call operations on Interface Definition Language (IDL) interfaces that are unknown at compile time. The interface repository (IFR) provides centralized persistent storage of IDL interfaces. Orbix programs can query the interface repository at runtime to obtain information about IDL definitions.

To verify that your Orbix server is registered in the Naming Service, enter the following command at the command prompt:

```
itadmin ns list
```

Your server name should appear within a list of all the registered servers. If it is not in the list, you must register your server.

To ensure your IFR is running and populated, you can view its contents using the following command:

```
itadmin ifr list
```

This command lists all the currently scoped names, such as interfaces and types.

You can use the itadmin command to view the IDL definition of one of the current scoped names, as follows:

```
itadmin ifr show current_scoped_name
```

In order to populate the interface repository with IDL definitions, run the IDL compiler with the `-R` option. For example, the following command populates the interface repository with the IDL definitions in `bank.idl`:

```
idl -I. -I$(ART_IDL_DIR)\omg -R= bank.idl
```

For more information on the Orbix utilities and command line, see Section IV of the *Orbix 2000 Administrators Guide*.

## Using VisiBroker for Java

VisiBroker is a complete CORBA 2.3 Object Request Broker (ORB) that supports the development, deployment, and management of distributed object applications across a variety of hardware platforms and operating systems. In addition to VisiBroker (the ORB), three other components are available with VisiBroker:

- Naming Service, which allows you to associate one or more logical names with an object implementation and to store those names in a namespace. It also lets client applications use this service to obtain an object reference using the logical name assigned to that object.
- Event Service, which provides a facility that separates the communication between objects. It provides a *supplier-consumer* communications model that allows multiple *supplier objects* to send data asynchronously to multiple *consumer objects* through an event channel.
- Gatekeeper, which runs on a Web server and enables client programs to locate and use objects that do not reside on the Web server and to receive callbacks, even when firewalls are being used. The Gatekeeper can also be used as an HTTP daemon, thereby eliminating the requirement for a separate HTTP server during the application development phase.

The BEA WebLogic Adapter for CORBA supports VisiBroker for Java Version 4.5 and communicates using IIOP version 1.1. Applications created with VisiBroker for Java can communicate with object implementations developed with VisiBroker for C++

VisiBroker requires the Java Development Kit (JDK) or the Java Runtime Environment (JRE). You can obtain these tools from the Sun Microsystems Web site (<http://java.sun.com/>). JRE version 1.2.2 or higher is required to run the VisiBroker Console. You must install the JRE before you install VisiBroker. However, VisiBroker supports any current version of Java for your applications.

The BEA WebLogic Adapter for CORBA requires that an IOR file be available to locate the reference to the Interface Repository. However, VisiBroker's IOR file is not automatically output to a file. Modify your IFR startup procedure to automatically output the startup IOR reference to a file. For example, use the following command to automatically output the IOR reference to `ir.ior`:

```
irep myIr >ir.ior
```

VisiBroker configuration and run-time requirements for using the BEA WebLogic Adapter for CORBA include:

- The PATH environment variable must point to the VisiBroker libraries used by BEA WebLogic Adapter for CORBA.
- The VisiBroker jar file must be in the class path.
- The IFR must be populated with the IDL of the objects for which you want to create Web services.
- The VisiBroker Naming Service and IFR must be running.
- The CORBA servers you are going to use must be running or be set up to start on demand.

For example, with VisiBroker you start the Interface Repository using the following command:

```
irep myIr >ir.ior
```

Here, *myIr* is the startup IOR reference.

With VisiBroker you start the naming service using the following command:

```
$start nameserv NS_name
```

Here, *NS\_name* is the name of the naming service.

To verify that your server is registered in the Naming Service and your IFR is loaded:

1. Choose Start→Programs→VisiBroker→VisiBroker Console.

2. Expand VisiBroker ORB Services.
3. Expand the Naming Services folder or the IFR folder.

A list of the Naming Service or IFR objects should appear in the right pane.

Alternately, you can enter the following command at a command prompt:

```
osfind
```

This command finds the name of the server running the VisiBroker Naming Service. It is usually the machine where VisiBroker is installed.

## Compiling and Running the Club Package

Perform the following steps to compile and run the club package in VisiBroker.

1. Open the file, BEA\_CORBA\_SAMPLES.zip, and within it, open the file beacorba.zip. Extract the contents of beacorba.zip to the directory where you installed VisiBroker.
2. Set the PATH for VisiBroker:

- In Windows, edit your PATH System environment variable through the control panel to add the following text:

```
visibroker_home\lib;visibroker_home\bin;
```

- On Unix, edit your path statement as follows in a script or shell:

```
export PATH=visibroker_home/lib;visibroker_home/bin;$PATH
```

In this and the following steps, *visibroker\_home* refers to VisiBroker home directory.

3. Start the ORB Smart Agent by running the *sagent* script.
4. Start the empty Interface Repository.
  - On Windows, open a command prompt, run the *irep bea* command, and pipe the output to an interface repository as follows:

```
visibroker_home\bea\club> irep bea>ir.txt
```

- On Unix, in a shell, run the *irep bea* command and pipe the output to an interface repository as follows:

```
visibroker_home/boa/club> irep bea>ir.txt
```

In these examples, `ir.txt` refers to any name for an interface repository, and `bea` is the naming context for the `irep` command:

5. Populate the repository with `clubmed.idl`:

- On Windows, enter the following command:

```
visibroker_home\boa\club>idl2ir -irep bea clubmed.idl
```

- On Unix, enter the following:

```
visibroker_home/boa/club> idl2ir -irep bea clubmed.idl
```

6. Generate Java stubs.

- On Windows, enter the following command:

```
visibroker_home>\boa\club> idl2java clubmed.idl
```

- On Unix, enter the following command:

```
visibroker_home>/boa/club> idl2java clubmed.idl
```

The previous commands create the directory structure

`visibroker_home\boa\club\boa\club` on Windows and

`visibroker_home/boa/club/boa/club` on Unix . These directories contain Java files.

7. Rename the `ClubServer.java.vb` to `ClubServer.java` and copy it into

`visibroker_home\boa\club\boa\club` directory on Windows and

`visibroker_home/boa/club/boa/club` on Unix.

8. To compile for VisiBroker, within the VisiBroker environment, rename

`JacORB1_4_1\boa\club\ClubServer.java.vb` to

`JacORB1_4_1\boa\club\ClubServer.java`

After renaming the file, copy `ClubServer.java` into the following directory:

```
visibroker_home/boa/club/boa/club
```

Here, `visibroker_home` is the VisiBroker home directory.

**Notes:** You create the directory above when you execute the `idl2java clubmed.idl` command.

9. Compile the classes:

- On Windows:

```
visibroker_home\bea\club\bea\club> vbjc -classpath  
visibroker_home\lib\vbjorb.jar:. bea\club\*.java bea\club  
\ClubMedPackage\*.java
```

- On Unix:

```
visibroker_home/bea/club> vbjc -classpath  
visibroker_home/lib/vbjorb.jar:. bea/club/*.java  
bea/club/ClubMedPackage/*.java
```

10. Start the server with object references:

- On Windows, enter the following commands in order:

```
visibroker_home\bea\club> nameserv  
visibroker_home\bea\club> vbj -DSVCnameroot=NameService  
-classpath visibroker_home\lib\vbjorb.jar:.  
bea.club.ClubServer -club
```

- On Unix, enter the following commands in order:

```
visibroker_home/bea/club> nameserv  
visibroker_home/bea/club> vbj -DSVCnameroot=NameService  
-classpath visibroker_home/lib/vbjorb.jar:.  
bea.club.ClubServer -club
```

For more information, refer to *VisiBroker for Java Installation Guide*, available at:

<http://info.borland.com/techpubs/books/vbj/vbj45/framesetindex.html>





# B Supported IDL Types

The BEA WebLogic Adapter for CORBA supports the following Interface Definition Language (IDL) base types:

- boolean
- char
- double
- float
- long
- long long
- octet
- short
- string
- unsigned long
- unsigned long long
- unsigned short
- wchar

The BEA WebLogic Adapter for CORBA supports the following IDL constructed types:

- array (where dimension is less than or equal to 2)
- enum

## **B** *Supported IDL Types*

---

- interface (as a return value, but not within a return value of another constructed type)
- sequence
- struct
- typedef
- union

# C Sample Files

This section describe the sample files delivered with the software. It includes the following topics:

- [Sample XML Request and Response Documents](#)
- [Definitions for ClubMed Object](#)

## Sample XML Request and Response Documents

This section includes the following sample request and response documents:

- [addGetReservation Request](#)
- [addGetReservation Responses](#)
- [AddReservation Request](#)
- [AddReservation Response](#)
- [addReservationComplex Request](#)
- [addReservationComplex Response](#)
- [addReservationWithOut Request](#)
- [addReservationWithOut Response](#)
- [cancelReservation Request](#)

- `cancelReservation` Response
- `getClub` Request
- `GetClubNames` Request
- `GetClubNames` Response
- `getClubNames4` Request
- `getClubNames4` Response
- `getClubNamesWithRef` Request
- `GetClubPrices` Request
- `GetClubPrices` Response
- `GetClubPricesAsArray` Request
- `GetClubPricesAsArray` Response
- `GetReservation` Request
- `GetReservation` Response
- `getReservationAsOut` Request
- `getReservationAsOut` Response
- `setCancel` Request
- `setCancel` Response
- `setClubPrices` Request
- `setClubPrices` Response
- `setClubPricesAsArray` Request
- `setClubPricesAsArray` Response
- `setStatus` Request
- `setStatus` Response
- `status` Request
- `status` Response

- [testOneDim Request](#)
- [testOneDim Response](#)
- [testTwoDim Request](#)
- [testTwoDim Response](#)
- [testTypes Request](#)
- [testTypes Response](#)
- [testUnion1-2 Request](#)
- [testUnion2-1 Request](#)
- [testUnion2-3 Request](#)
- [testUnion3-PROCESSED Request](#)
- [testUnion4-1 Request](#)
- [testUnion4-default Request](#)

## addGetReservation Request

### Listing C-1 addGetReservation Request

---

```
<bea.club.ClubMed.addGetReservation>
  <club>BAMBU</club>
  <resvData>
    <fname>Bill</fname>;
    <lname>Wales</lname>
    <weekDate>Monday</weekDate>
    <street>Big Bungalow</street>
    <city>High End</city>
    <state>WA</state>
    <zip>99990</zip>
    <phone>123-123-3456</phone>
    <totalFare>5000.0</totalFare>
    <partyAdults>2</partyAdults>
    <partyChildren>2</partyChildren>
    <date>2001-09-21</date>
  </resvData>
</addGetReservation>
```

```
</resvData>  
</bea.club.ClubMed.addGetReservation>
```

---

# addGetReservation Response

## Listing C-2 addGetReservation Response

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>  
<bea.club.ClubMed.addGetReservation>  
  <return>1002</return>  
  <resvData>  
    <fname>Bill</fname>  
    <lname>Wales</lname>  
    <weekDate>Monday</weekDate>  
    <street>Big Bungalow</street>  
    <city>High End</city>  
    <state>WA</state>  
    <zip>99990</zip>  
    <phone>123-123-3456</phone>  
    <totalFare>5000.0</totalFare>  
    <partyAdults>2</partyAdults>  
    <partyChildren>2</partyChildren>  
    <date>2001-09-21</date>  
    <resv>1000</resv>  
  </resvData>  
</bea.club.ClubMed.addGetReservation>
```

---

# AddReservation Request

## Listing C-3 AddReservation Request

---

```
<bea.club.ClubMed.addReservation>  
  <club>BAMBU</club>  
  <resvData>  
    <fname>Bill</fname>;  
    <lname>Wales</lname>  
    <weekDate>Monday</weekDate>
```

```
<address>Big Bungalow</address>
<city>High End</city>
<state>WA</state>
<zip>99990</zip>
<phone>123-123-3456</phone>
<totalFare>5000.0</totalFare>
<partyAdults>2</partyAdults>
<partyChildren>2</partyChildren>
<date>2001-09-21</date>
</resvData>
</bea.club.ClubMed.addReservation>
```

---

## AddReservation Response

### Listing C-4 AddReservation Response

---

```
<?xml version="1.0"?>
<bea.club.ClubMed.addReservation>
  <return>1001</return>
</bea.club.ClubMed.addReservation>
```

---

## addReservationComplex Request

### Listing C-5 addReservationComplex Request

---

```
<bea.club.ClubMed.addReservationComplex>
  <club>BAMBU</club>
  <resvData>
    <addr>
      <fname>Bill</fname>;
      <lname>Wales</lname>
      <street>Big Bungalow</street>
      <city>High End</city>
      <state>WA</state>
      <zip>99990</zip>
      <phone>123-123-3456</phone>
      <prices>
```

```
<bea.club.pricesStruct>
  <date>2002-11-23</date>
  <adultFare>88</adultFare>
  <childFare>50</childFare>
</bea.club.pricesStruct>
<bea.club.pricesStruct>
  <date>2002-11-24</date>
  <adultFare>88</adultFare>
  <childFare>50</childFare>
</bea.club.pricesStruct>
<bea.club.pricesStruct>
  <date>2002-11-25</date>
  <adultFare>88</adultFare>
  <childFare>50</childFare>
</bea.club.pricesStruct>
<bea.club.pricesStruct>
  <date>2002-11-26</date>
  <adultFare>88</adultFare>
  <childFare>50</childFare>
</bea.club.pricesStruct>
</prices>
</addr>
<totalFare>5000.0</totalFare>
<partyAdults>2</partyAdults>
<partyChildren>2</partyChildren>
<date>2001-09-21</date>
<clubs>
  <item>BAMBU</item>
  <item>RIU PALACE</item>
</clubs>
</resvData>
</bea.club.ClubMed.addReservationComplex>
```

---

## addReservationComplex Response

### Listing C-6 addReservationComplex Response

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.addReservationComplex>
  <return>1000</return>
  <resvData>
    <addr>
      <fname>Bill</fname>
```



```
<lname>Wales</lname>
<street>Big Bungalow</street>
<city>High End</city>
<state>WA</state>
<zip>99990</zip>
<phone>123-123-3456</phone>
<prices>
  <bea.club.pricesArray>
    <date>2002-11-23</date>
    <adultFare>88</adultFare>
    <childFare>50</childFare>
  </bea.club.pricesArray>
  <bea.club.pricesArray>
    <date>2002-11-24</date>
    <adultFare>88</adultFare>
    <childFare>50</childFare>
  </bea.club.pricesArray>
  <bea.club.pricesArray>
    <date>2002-11-25</date>
    <adultFare>88</adultFare>
    <childFare>50</childFare>
  </bea.club.pricesArray>
  <bea.club.pricesArray>
    <date>2002-11-26</date>
    <adultFare>88</adultFare>
    <childFare>50</childFare>
  </bea.club.pricesArray>
</prices>
</addr>
<totalFare>5000.0</totalFare>
<partyAdults>2</partyAdults>
<partyChildren>2</partyChildren>
<date>2001-09-21</date>
<clubs>
  <item>BAMBU</item>
  <item>RIU PALACE</item>
</clubs>
</resvData>
</bea.club.ClubMed.addReservationComplex>
```

---

## **addReservationWithOut Request**

### **Listing C-7 addReservationWithOut Request**

---

```
<bea.club.ClubMed.addReservationWithOut>
  <club>BAMBU</club>
  <resvData>
    <fname>Bill</fname>;
    <lname>Wales</lname>
    <weekDate>Monday</weekDate>
    <street>Big Bungalow</street>
    <city>High End</city>
    <state>WA</state>
    <zip>99990</zip>
    <phone>123-123-3456</phone>
    <totalFare>5000.0</totalFare>
    <partyAdults>2</partyAdults>
    <partyChildren>2</partyChildren>
    <date>2001-09-21</date>
    <resv>0</resv>
  </resvData>
</bea.club.ClubMed.addReservationWithOut>
```

---

## **addReservationWithOut Response**

### **Listing C-8 addReservationWithOut Response**

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.addReservationWithOut>
  <num>1004</num>
</bea.club.ClubMed.addReservationWithOut>
```

---

## cancelReservation Request

### Listing C-9 cancelReservation Request

---

```
<bea.club.ClubMed.cancelReservation>  
  <resv>1000</resv>  
</bea.club.ClubMed.cancelReservation>
```

---

## cancelReservation Response

### Listing C-10 cancelReservation Response

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>  
<bea.club.ClubMed.cancelReservation>  
  <return>false</return>  
</bea.club.ClubMed.cancelReservation>
```

## getClub Request

### Listing C-11 GetClub Request

---

```
<bea.club.Agency.getClub>  
</bea.club.Agency.getClub>
```

---

## GetClubNames Request

### Listing C-12 GetClubNames Request

---

```
<bea.club.ClubMed.getClubNames>  
</bea.club.ClubMed.getClubNames>
```

---

## GetClubNames Response

### Listing C-13 GetClubNames Response

---

```
<?xml version="1.0"?>  
<bea.club.ClubMed.getClubNames>  
  <bea.club.clubNamesSeq>  
    <return>BAMBU</return>  
    <return>NAEBO</return>  
    <return>RIO PALACE</return>  
  </bea.club.clubNamesSeq>  
</bea.club.ClubMed.getClubNames>
```

---

## getClubNames4 Request

### Listing C-14 getClubNames4 Request

---

```
<bea.club.ClubMed.getClubNames4>  
</bea.club.ClubMed.getClubNames4>
```

---

## getClubNames4 Response

### Listing C-15 getClubNames4 Response

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.getClubNames4>
  <return>
    <item>BAMBU</item>
    <item>NAEBO</item>
    <item>RIU PALACE</item>
  </return>
</bea.club.ClubMed.getClubNames4>
```

---

## getClubNamesWithRef Request

### Listing C-16 GetClubNamesWithRef Request

---

```
<bea.club.ClubMed.getClubNames
  ref="IOR:0000000000000001949444c3a6265612f636c75622f436c75624d6
5643a312e300000000000000001000000000000078000102000000000e313
7322e31392e32352e323131000c460000001c00564201000000022f0020200
000000400000001000001d738b66b92000000035649530300000005000404010f
00000000000000000000080000000056495300000000010000001400000000000
10001000000000001010900000000">
</bea.club.ClubMed.getClubNames>
```

---

## GetClubPrices Request

### Listing C-17 GetClubPrices Request

---

```
<bea.club.ClubMed.getClubPrices>
  <club>BAMBU</club>
</bea.club.ClubMed.getClubPrices>
```

## GetClubPrices Response

### Listing C-18 GetClubPrices Response

---

```
<?xml version="1.0"?>
<bea.club.ClubMed.getClubPrices>
  <bea.club.pricesSeq>
    <bea.club.pricesStruct>
      <date>2002-11-23</date>
      <adultFare>88</adultFare>
      <childFare>50</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-24</date>
      <adultFare>88</adultFare>
      <childFare>50</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-25</date>
      <adultFare>80</adultFare>
      <childFare>45</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-26</date>
      <adultFare>80</adultFare>
      <childFare>45</childFare>
    </bea.club.pricesStruct>
  </bea.club.pricesSeq>
</bea.club.ClubMed.getClubPrices>
```

---

## GetClubPricesAsArray Request

### Listing C-19 GetClubPricesAsArray Request

---

```
<bea.club.ClubMed.getClubPricesAsArray>
  <club>BAMBU</club>
</bea.club.ClubMed.getClubPricesAsArray>
```

---

## GetClubPricesAsArray Response

### Listing C-20 GetClubPricesAsArray Response

---

```
<?xml version="1.0"?>
<bea.club.ClubMed.getClubPricesAsArray>
  <bea.club.pricesArray>
    <bea.club.pricesStruct>
      <date>2002-11-23</date>
      <adultFare>88</adultFare>
      <childFare>50</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-24</date>
      <adultFare>88</adultFare>
      <childFare>50</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-25</date>
      <adultFare>80</adultFare>
      <childFare>45</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-26</date>
      <adultFare>80</adultFare>
      <childFare>45</childFare>
    </bea.club.pricesStruct>
  </bea.club.pricesArray>
</bea.club.ClubMed.getClubPricesAsArray>
```

---

## GetReservation Request

### Listing C-21 GetReservation Request

---

```
<bea.club.ClubMed.getReservation>  
  <resv>1001</resv>  
</bea.club.ClubMed.getReservation>
```

---

## GetReservation Response

### Listing C-22 GetReservation Response

---

```
<?xml version="1.0"?>  
<bea.club.ClubMed.getReservation>  
  <bea.club.resvStruct>  
    <fname>Bill</fname>  
    <lname>Wales</lname>  
    <weekDate>Monday</weekDate>  
    <address>Big Bungalow</address>  
    <city>High End</city>  
    <state>WA</state>  
    <zip>99990</zip>  
    <phone>123-123-3456</phone>  
    <totalFare>5000.0</totalFare>  
    <partyAdults>2</partyAdults>  
    <partyChildren>2</partyChildren>  
    <date>2001-09-21</date>  
  </bea.club.resvStruct>  
</bea.club.ClubMed.getReservation>
```

---



## getReservationAsOut Request

### Listing C-23 getReservationAsOut Request

---

```
<bea.club.ClubMed.getReservationAsOut>
  <resv>1000</resv>
</bea.club.ClubMed.getReservationAsOut>
```

---

## getReservationAsOut Response

### Listing C-24 getReservationAsOut Response

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.getReservationAsOut>
  <resvData>
    <addr>
      <fname>Bill</fname>
      <lname>Wales</lname>
      <street>Big Bungalow</street>
      <city>High End</city>
      <state>WA</state>
      <zip>99990</zip>
      <phone>123-123-3456</phone>
    <prices>
      <bea.club.pricesArray>
        <date>2002-11-23</date>
        <adultFare>88</adultFare>
        <childFare>50</childFare>
      </bea.club.pricesArray>
      <bea.club.pricesArray>
        <date>2002-11-24</date>
        <adultFare>88</adultFare>
        <childFare>50</childFare>
      </bea.club.pricesArray>
      <bea.club.pricesArray>
        <date>2002-11-25</date>
        <adultFare>88</adultFare>
        <childFare>50</childFare>
      </bea.club.pricesArray>
      <bea.club.pricesArray>
```

```
        <date>2002-11-26</date>
        <adultFare>88</adultFare>
        <childFare>50</childFare>
    </bea.club.pricesArray>
</prices>
</addr>
<totalFare>5000.0</totalFare>
<partyAdults>2</partyAdults>
<partyChildren>2</partyChildren>
<date>2001-09-21</date>
<clubs>
    <item>BAMBU</item>
    <item>RIU PALACE</item>
</clubs>
</resvData>
</bea.club.ClubMed.getReservationAsOut>
```

---

## setCancel Request

### Listing C-25 setCancel Request

---

```
<bea.club.ClubMed.setCancel>
  <resv>1000</resv>
  <cancel>false</cancel>
</bea.club.ClubMed.setCancel>
```

---

## setCancel Response

### Listing C-26 setCancel Response

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.setCancel/>
```

---

## setClubPrices Request

### Listing C-27 setClubPrices Request

---

```
<bea.club.ClubMed.setClubPrices>
  <club>BAMBU</club>
  <prices>
    <bea.club.pricesStruct>
      <date>2002-11-23</date>
      <adultFare>88</adultFare>

      <childFare>50</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-24</date>
      <adultFare>88</adultFare>
      <childFare>50</childFare>
    </bea.club.pricesStruct>
  </prices>
</bea.club.ClubMed.setClubPrices>
```

---

## setClubPrices Response

### Listing C-28 setClubPrices Response

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.setClubPrices/>
```

---

## setClubPricesAsArray Request

### Listing C-29 setClubPricesAsArray Request

---

```
<bea.club.ClubMed.setClubPricesAsArray>
  <club>BAMBU</club>
```

```
<prices>
  <bea.club.pricesStruct>
    <date>2002-11-23</date>
    <adultFare>88</adultFare>
    <childFare>50</childFare>
  </bea.club.pricesStruct>
  <bea.club.pricesStruct>
    <date>2002-11-24</date>
    <adultFare>88</adultFare>
    <childFare>50</childFare>
  </bea.club.pricesStruct>
  <bea.club.pricesStruct>
    <date>2002-11-24</date>
    <adultFare>88</adultFare>
    <childFare>50</childFare>
  </bea.club.pricesStruct>
  <bea.club.pricesStruct>
    <date>2002-11-24</date>
    <adultFare>88</adultFare>
    <childFare>50</childFare>
  </bea.club.pricesStruct>
</prices>
</bea.club.ClubMed.setClubPricesAsArray>
```

---

## setClubPricesAsArray Response

### Listing C-30 setClubPricesAsArray Response

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.setClubPricesAsArray/>
```

---

## setStatus Request

### Listing C-31 setStatus Request

---

```
<bea.club.ClubMed.setStatus>
  <resv>1000</resv>
```

```
<cod>CANCELLED</cod>
</bea.club.ClubMed.setStatus>
```

---

## setStatus Response

### Listing C-32 setStatus Response

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.setStatus/>
```

---

## status Request

### Listing C-33 status Request

---

```
<bea.club.ClubMed.status>
  <resv>1000</resv>
</bea.club.ClubMed.status>
```

---

## status Response

### Listing C-34 status Response

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.status>
  <return>PROCESSED</return>
</bea.club.ClubMed.status>
```

---

## testOneDim Request

### Listing C-35 testOneDim Request

---

```
<bea.club.ClubMed.testOneDim>
  <par>
    <item>1</item>
    <item>2</item>
    <item>3</item>
  </par>
</bea.club.ClubMed.testOneDim>
```

---

## testOneDim Response

### Listing C-36 testOneDim Response

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.testOneDim>
  <par>
    <item>1</item>
    <item>2</item>
    <item>3</item>
  </par>
</bea.club.ClubMed.testOneDim>
```

---

## testTwoDim Request

### Listing C-37 testTwoDim Request

---

```
<bea.club.ClubMed.testTwoDim>
  <par>
    <bea.club.twoDim_row>
      <item>one</item>
      <item>two</item>
```

```
        <item>three</item>
        <item>four</item>
        <item>five</item>
    </bea.club.twoDim_row>
    <bea.club.twoDim_row>
        <item>one</item>
        <item>two</item>
        <item>three</item>
        <item>four</item>
        <item>five</item>
    </bea.club.twoDim_row>
    <bea.club.twoDim_row>
        <item>one</item>
        <item>two</item>
        <item>three</item>
        <item>four</item>
        <item>five</item>
    </bea.club.twoDim_row>
</par>
</bea.club.ClubMed.testTwoDim>
```

---

## testTwoDim Response

### Listing C-38 testTwoDim Response

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.testTwoDim>
  <par>
    <bea.club.twoDim_row>
      <item>one</item>
      <item>two</item>
      <item>three</item>
      <item>four</item>
      <item>five</item>
    </bea.club.twoDim_row>
    <bea.club.twoDim_row>
      <item>one</item>
      <item>two</item>
      <item>three</item>
      <item>four</item>
      <item>five</item>
    </bea.club.twoDim_row>
    <bea.club.twoDim_row>
```

```
<item>one</item>
<item>two</item>
<item>three</item>
<item>four</item>
<item>five</item>
</bea.club.twoDim_row>
</par>
</bea.club.ClubMed.testTwoDim>
```

---

## testTypes Request

### Listing C-39 testTypes Request

---

```
<bea.club.ClubMed.testTypes>
  <all>
    <t1>string10</t1>
    <t2>1000</t2>
    <t3>1000999</t3>
    <t4>12345</t4>
    <t5>99999999</t5>
    <t6>1.234</t6>
    <t7>98765.1234</t7>
    <t8>c</t8>
    <t9>false</t9>
    <t10>34</t10>
    <t11>w</t11>
    <t14>123456789012899</t14>
    <t15>12345678901289999</t15>
  </all>
</bea.club.ClubMed.testTypes>
```

---

## testTypes Response

### Listing C-40 testTypes Response

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.testTypes>
```



```
<all>
  <t1>string10</t1>
  <t2>1000</t2>
  <t3>1000999</t3>
  <t4>12345</t4>
  <t5>99999999</t5>
  <t6>1.234</t6>
  <t7>98765.1234</t7>
  <t8>c</t8>
  <t9>false</t9>
  <t10>34</t10>
  <t11>w</t11>
  <t14>123456789012899</t14>
  <t15>12345678901289999</t15>
</all>
</bea.club.ClubMed.testTypes>
```

---

## testUnion1-2 Request

### Listing C-41 testUnion1-2 Request

---

```
<bea.club.ClubMed.testUnion1>
  <moon>
    <case.2>123456</case.2>
  </moon>
</bea.club.ClubMed.testUnion1>
```

---

## testUnion2-1 Request

### Listing C-42 testUnion2-1 Request

---

```
<bea.club.ClubMed.testUnion2>
  <moon>
    <case.1>blah</case.1>
  </moon>
</bea.club.ClubMed.testUnion2>
```

---

## testUnion2-3 Request

### Listing C-43 testUnion2-3 Request

---

```
<bea.club.ClubMed.testUnion2>
  <moon>
    <case.3>
      <t1>string10</t1>
      <t2>1000</t2>
      <t3>1000999</t3>
      <t4>12345</t4>
      <t5>99999999</t5>
      <t6>1.234</t6>
      <t7>98765.1234</t7>
      <t8>c</t8>
      <t9>false</t9>
      <t10>34</t10>
      <t11>w</t11>
      <t14>123456789012899</t14>
      <t15>12345678901289999</t15>
    </case.3>
  </moon>
</bea.club.ClubMed.testUnion2>
```

---

## testUnion3-PROCESSED Request

### Listing C-44 testUnion3-PROCESSED Request

---

```
<bea.club.ClubMed.testUnion3>
  <moon>
    <case.PROCESSED>123456</case.PROCESSED>
  </moon>
</bea.club.ClubMed.testUnion3>
```

---

## testUnion4-1 Request

### Listing C-45 testUnion4-1 Request

---

```
<bea.club.ClubMed.testUnion4>
  <moon>
    <case.1>blah</case.1>
  </moon>
</bea.club.ClubMed.testUnion4>
```

---

## testUnion4-default Request

### Listing C-46 testUnion4-default Request

---

```
<bea.club.ClubMed.testUnion4>
  <moon>
    <case.default>true</case.default>
  </moon>
</bea.club.ClubMed.testUnion4>
```

---

## Definitions for ClubMed Object

The BEA WebLogic Adapter for CORBA provides the sample ClubMed object to enable you to test the adapter.

## JacORB.properties File

The following listing reflects the JacORB.properties file after you have edited it according to the instructions in [“Building and Running the JacORB Request Broker”](#) in [“Using JacORB”](#) in [Appendix A](#), [“Using CORBA Implementations With the Adapter.”](#)

### Listing C-47 JacORB.properties File

---

```
##
## JacORB configuration options
##

#####
#                                     #
#   Initial references configuration   #
#                                     #
#####

#
# URLs where IORs are stored (used in orb.resolve_initial_service())
# DO EDIT these! (Only those that you are planning to use,
# of course ;-).
#
# The ORBInitRef references are created on ORB startup time. In the
# cases of the services themselves, this may lead to exceptions being
# displayed (because the services aren't up yet). These exceptions
# are handled properly and cause no harm!

#ORBInitRef.NameService=corbaloc::160.45.110.41:38693/StandardNS/NameServer%2DPOA/_root
ORBInitRef.NameService=file:/d:/JacORB1_4_beta4/bea/ns_ref.txt
#ORBInitRef.NameService=http://www.x.y.z/~user/NS_Ref
#ORBInitRef.TradingService=http://www.x.y.z/~user/TraderRef

# JacORB-specific URLs
jacorb.ProxyServerURL=http://www.x.y.z/~user/Applicator_Ref

#####
#                                     #
#   ORB version number output        #
#                                     #
#####
```

```
# if on, the ORB's version number is printed
# any time the ORB is initialized
jacorb.orb.print_version=on

#####
#                                     #
#   Debug output configuration      #
#                                     #
#####

# use (java) jacob.util.CAD to generate an appropriate
# verbosity level
# 0 = off
# 1 = important messages and exceptions
# 2 = informational messages and exceptions
# >= 3 = debug-level output (may confuse the unaware user :-)
jacorb.verbosity=1

# where does output go? Terminal is default
#jacorb.logfile=LOGFILEPATH

# hexdump outgoing messages
jacorb.debug.dump_outgoing_messages=off

# hexdump incoming messages
jacorb.debug.dump_incoming_messages=off

#####
#                                     #
#   WARNING: The following properties should   #
#   only be edited by the expert user. They   #
#   can be left untouched in most cases!      #
#                                     #
#####

#####
#                                     #
#   Basic ORB Configuration      #
#                                     #
#####

# the GIOP minor version number to use for newly created IORs
jacorb.giop_minor_version=2

# number of retries if connection cannot directly be established
```

```
jacorb.retries=5

# how many msecs. do we wait between retries
jacorb.retry_interval=500

# size of network buffers for outgoing messages
jacorb.outbuf_size=2048

# log2 of maximum buffer size managed by the internal
# buffer manager.
#
# This is NOT the maximum buffer size that
# can be used, but just the largest size of buffers that
# will be kept and managed. This value will be added to
# an internal constant of 5, so the real value in bytes
# is 2**(5+maxManagedBufSize-1). You only need to increase this
# value if you are dealing with LOTS of LARGE data structures.
# You may decrease it to make the buffer manager release large
# buffers immediately rather than keeping them for later
# reuse.
jacorb.maxManagedBufSize=18

# client-side timeout, set no non-zero to stop blocking
# after so many msecs.
#jacorb.connection.client_timeout=0

# max time a server keeps a connection open if nothing happens
#jacorb.connection.server_timeout=10000

#jacorb.reference_caching=off

#
# The following property specifies the class which is used for
# reference caching. WeakHashtable uses WeakReferences, so entries
# get gc'ed if only the Hashtable has a reference to them. This
# is useful if you have many references to short-living non-persistent
# CORBA objects. It is only available for java 1.2 and above.
#
# On the other hand the standard Hashtable keeps the references until
# they are explicitly deleted by calling _release(). This is useful
# for persistent and long-living CORBA objects.
#
#jacorb.hashtable_class=org.jacorb.util.WeakHashtable
#
jacorb.hashtable_class=java.util.Hashtable

# use GIOP 1.2 byte order markers (since CORBA 2.4-5)
jacorb.use_bom=off
```

```
# add additional IIOP 1.0 profiles even if we are using IIOP 1.2
jacorb.giop.add_1_0_profiles=off

#####
#                                     #
#          Socket Factories          #
#                                     #
#####

# A factory design pattern is used for the creation of sockets and server
# sockets.
# The jacob.net.socket_factory property can be used to configure
# a socket factory that must implement the operations defined in the
# interface org.jacob.net.factory.SocketFactory.
# The jacob.net.server_socket_factory property can be used to configure a
# server socket factory that must implement the operations defined in the
# interface org.jacob.net.factory.ServerSocketFactory.
#
#jacob.net.socket_factory=org.jacob.net.factory.DefaultSocketFactory
#jacob.net.server_socket_factory=org.jacob.net.factory.DefaultServerSocketFactory
#
# An additional socket factory is supported that allows for the configuration
# of maximum and minimum port numbers that can be used. This can be used to
# enable firewall traversal via a fixed port range. To use this socket factory
# configure the following two properties.
#
#jacob.net.socket_factory.port.min
#jacob.net.socket_factory.port.max

#####
#                                     #
#          BiDirectional GIOP        #
#                                     #
#####

# uncomment this initializer if you want to use BiDirectional GIOP

#org.omg.PortableInterceptor.ORBInitializerClass.bidir_init=org.jacob.net.connection.BiDirConnectionInitializer

#####
#                                     #
#          Proxy address in IOR      #
#                                     #
#####

#
```

```
# with these two properties it is possible to
# tell the ORB what IP/port IORs should contain,
# if the ServerSockets IP/port can't be used
# (e.g. for traffic through a firewall).
#
# WARNING: this is just "dumb" replacing, so you
# have to take care of your configuration!
#

#jacorb.ior_proxy_host=1.2.3.4
#jacorb.ior_proxy_port=4711

#####
#                               #
#   The Object Adapter Internet Address   #
#                               #
#####

# IP address on multi-homed host (this gets encoded in
# object references). NOTE: Addresses like 127.0.0.X
# will only be accessible from the same machine!
#OAIAddr=1.2.3.4
#OAPort=4711

#####
#                               #
#   Appligator Configuration   #
#                               #
#####
# if your applets don't need appligator, switch this off
jacorb.use_appligator=off

#####
#                               #
#   Default Interceptors   #
#   Please leave them in!   #
#                               #
#####
org.omg.PortableInterceptor.ORBInitializerClass.standard_init=org.jacorb.orb.st
andardInterceptors.IORInterceptorInitializer

#####
#                               #
#   Implementation Repository Configuration   #
```



```
#                                     #
#####
# Switch off to avoid contacting the ImR on every server start-up
jacorb.use_imr=off

# if set to "on", servers that don't already have an entry on their
# first call to the imr, will get automatically registered. Otherwise,
# an UnknownServer exception is thrown.
jacorb.imr.allow_auto_register=off

# if set to "on", the imr will try to "ping" every object reference,
# that it is going to return. If the reference is not alive, TRANSIENT
# is thrown.
jacorb.imr.check_object_liveness=off

ORBInitRef.ImplementationRepository=http://www.x.y.z/~user/ImR_Ref

jacorb.imr.table_file=Z:\table.dat
jacorb.imr.backup_file=z:\backup.dat
jacorb.imr.ior_file=/home/bwana/brose/public_html/ImR_Ref
jacorb.imr.timeout=
jacorb.imr.no_of_poas=
jacorb.imr.no_of_servers=

# how many millis should the imr wait, until a connection from an
# application client is terminated. Default is 2000.
jacorb.imr.connection_timeout=2000

# the implementation name, should be set to a different
# name in the code of persistent servers
jacorb.implname=StandardImplName

#
# This is supposed to be a generic startup string for everything
# that calls Runtime.exec(). Might be replaced by jaco[.bat].
#
jacorb.java_exec=java -Dorg.omg.CORBA.ORBClass=org.jacorb.orb.ORB
-Dorg.omg.CORBA.ORBSingletonClass=org.jacorb.orb.ORBSingleton

#####
#                                     #
#   SSL Configuration   #
#                                     #
#####

#
# The port number used by SSL, will be dynamically assigned
```

```
# by default
#

#OASLPort=4711

# This interceptor must be set if programs need access to
# certificates using the CORBA Security API, SSL works also
# without this interceptor

#org.omg.PortableInterceptor.ORBInitializerClass.ForwardInit=org.jacorb.security.ssl.SecurityServiceInitializer

# qualified classname of access decision object
jacorb.security.access_decision=org.jacorb.security.level2.AccessDecisionImpl

# list of qualified classnames of principal authenticator objects,
# separated by commas (no whitespaces!). The first entry (that can
# be successfully created) will be available through the
# principal_authenticator property.
jacorb.security.principal_authenticator=org.jacorb.security.level2.PrincipalAuthenticatorImpl

# the qualified classname of the ssl socket factory class
#jacorb.ssl.socket_factory=org.jacorb.security.ssl.sun_jsse.SSLSocketFactory
jacorb.ssl.socket_factory=org.jacorb.security.ssl.iaik.SSLSocketFactory

# the qualified classname of the ssl server socket factory class
#jacorb.ssl.server_socket_factory=org.jacorb.security.ssl.sun_jsse.SSLServerSocketFactory
jacorb.ssl.server_socket_factory=org.jacorb.security.ssl.iaik.SSLServerSocketFactory

# exchange ssl client server roles to enforce client authentication, but
# attention: this causes problems with peers that not prepared to handle
# this role change
jacorb.security.change_ssl_roles=off

# IIOP/SSL parameters (numbers are hex values, without the leading "0x"):
# NoProtection = 1
# EstablishTrustInClient = 40
# EstablishTrustInTarget = 20
# mutual authentication = 60
# please see the programming guide for more explanation

jacorb.security.support_ssl=off

jacorb.security.ssl.client.supported_options=0
jacorb.security.ssl.client.required_options=0
```

```
jacorb.security.ssl.server.supported_options=0
jacorb.security.ssl.server.required_options=0

#
# If set, the following two values will be placed in the IOR, if
# "corbaloc:ssliop" ssliop.
#
# If not set, only EstablishTrustInTarget is used for both supported
# and required options. EstablishTrustInClient is not set, and the
# rest of the Association Options aren't currently used anyway.
#jacorb.security.ssl.corbaloc_ssliop.supported_options=0
#jacorb.security.ssl.corbaloc_ssliop.required_options=0

# The name and location of the keystore. This may be absolute or
# relative to the home directory.
#
# NOTE (for Sun JSSE users): The "javax.net.ssl.trustStore[Password]"
# properties don't seem to take effect, so you may want to add trusted
# certificates to "normal" keystores. In this case, please set the
# property "jacorb.security.jsse.trustees_from_ks" is to "on", so trusted
# certificates are taken from the keystore instead of a dedicated
# truststore.
jacorb.security.keystore=
jacorb.security.keystore_password=

#
# IAIK specific settings
#

# files with public key certs of trusted CAs
#
# WARNING: If no CA certs are present, the IAIK chain verifier will
# accept ALL otherwise valid chains!
#
jacorb.security.trustees=

# the name of the default key alias to look up in the keystore
jacorb.security.default_user=
jacorb.security.default_password=

# have iaiks ssl classes print debug output to stdout
jacorb.security.iaik_debug=off

#
# Sun JSSE specific settings
#
# Use the keystore to take trusted certs from.
```

```
jacorb.security.jsse.trustees_from_ks=off

# A comma-separated (no whitespaces!) list of cipher suite names. See
# the JSSE docs on how to obtain the correct cipher suite strings
jacorb.security.ssl.server.cipher_suites=
jacorb.security.ssl.client.cipher_suites=

#####
#                               #
#   POA Configuration         #
#                               #
#####

# displays a GUI monitoring tool for servers
jacorb.poa.monitoring=off

# thread pool configuration for request processing
jacorb.poa.thread_pool_max=20
jacorb.poa.thread_pool_min=5

# if set, request processing threads in the POA
# will run at this priority. If not set or invalid,
# MAX_PRIORITY will be used.
#jacorb.poa.thread_priority=

# size of the request queue, clients will receive Corba.TRANSIENT
# exceptions if load exceeds this limit
jacorb.poa.queue_max=100

#####
#                               #
#   Trader configuration, please see #
#   src/trading/README.PROPERTIES for #
#   explanation                     #
#                               #
#####

jtrader.util.max_threads=10
jtrader.util.min_threads=1
jtrader.util.query_timeout=5000
jtrader.impl.cache_max=100

# boolean values, e.g. true / false
#jtrader.modifiable_properties=
#jtrader.dynamic_properties=
#jtrader.proxy_offers=
```

```
jtrader.debug=false
jtrader.debug_verbosity=3

#integer values
jtrader.def_search_card=
jtrader.max_search_card=
jtrader.def_match_card=
jtrader.max_match_card=
jtrader.def_return_card=
jtrader.max_return_card=
jtrader.max_list=
jtrader.def_hop_count=
jtrader.max_hop_count=

#FollowOptions
#always=2
#if_no_local=1
#local_only=0
jtrader.def_follow_policy=
jtrader.max_follow_policy=
jtrader.max_link_follow_policy=

# any other custom properties can be added here.
# These are available through the API (call
# jacorb.orb.Environment.getProperty())
```

---

## Build XML

This file enables you to build the sample JacORB application referred to in [“Building and Running the JacORB Request Broker”](#) in [“Using JacORB”](#) in [Appendix A, “Using CORBA Implementations With the Adapter.”](#)

### Listing C-48 Build XML

---

```
<?xml version="1.0"?>

<project name="bea" default="all" basedir=" ../.. ">

  <!-- ===== -->
  <!--           build file           -->
  <!-- ===== -->
```

```
<target name="init">
  <property name="name" value="club"/>
  <property name="dirs.base" value="${basedir}"/>
  <property name="classdir" value="${dirs.base}/classes"/>
  <property name="lib" value="${dirs.base}/lib"/>
  <property name="include" value="${dirs.base}/idl"/>
  <property name="idlflags" value="-I${include}/omg -ir -d
${dirs.base}/bea/${name}/generated"/>
</target>

<target name="all" depends="init,idl">
  <jived srcdir="${dirs.base}/bea/${name}/generated"
        destdir="${classdir}"
        includes="**/*.java"
        />

  <jived srcdir="${dirs.base}/bea/${name}"
        destdir="${classdir}"
        includes="*.java"
        />
</target>

<target name="idl" depends="init">
  <java classname="org.jacorb.idl.parser"
        fork="yes"
        classpath="${lib}/idl.jar;${java.class.path}">

    <are line="${divulges}
    ${dirs.base}/bea/${name}/clubmed.idl"/>
  </java>
</target>

<target name="clean" depends="init">
  <delete dir="${cloister}/bea/${name}"/>
  <delete dir="${dirs.base}/bea/${name}/generated"/>
</target>

</project>
```

---

## ClubMed.IDL File

The clubmed.idl file displays the Interface Repository in IDL.

**Listing C-49**

```

// ClubMed.idl
module bea
{
module club
{
    struct typesStruct
    {
        string<10>          t1;
        short               t2; /* 16 bit */
        long                t3; /* 32 bit */
        unsigned short      t4;
        unsigned long        t5;
        float               t6;
        double              t7;
        char                t8;
        boolean             t9;
        octet               t10; /* byte */
        wchar               t11;
        /* restring          t12; */
        /* wstring<10>       t13; */
        long long           t14; /* 64 bit */
        unsigned long long  t15;
        /* long double       t16; */
        /* fixed<5,2>        t17; */
    };

    typedef long    oneDim[3];
    typedef string twoDim[3][5];

    struct pricesStruct
    {
        string date;
        short adultFare;
        short childFare;
    };

    typedef sequence <string,4> clubNamesSeq4;

    typedef sequence <pricesStruct> pricesSeq;

    typedef sequence <string> clubNamesSeq;

    typedef long resvNo;

    typedef pricesStruct pricesArray[4];

```

```
struct resvStruct
{
    string fname;
    string lname;
    string weekDate;
    string street;
    string city;
    string state;
    string zip;
    string phone;
    string totalFare;
    short partyAdults;
    short partyChildren;
    string date;
    resvNo resv;
};

struct address
{
    string fname;
    string lname;
    string street;
    string city;
    string state;
    string zip;
    string phone;
    pricesArray prices;
};

struct resv Complex Struct
{
    address addr;
    string totalFare;
    short partyAdults;
    short partyChildren;
    string date;
    clubNamesSeq clubs;
};

enum code { OPENED, PROCESSED, CANCELED};

union myUnion4 switch(long) {
    case 1: string name;
    case 2: long num;
    case 3: typesStruct str;
    default: boolean b;
};
```



```
union myUnion1 switch(char) {
    case '1': string name;
    case '2': long  num;
    case '3': typesStruct str;
};

union myUnion2 switch(short) {
    case 1: string name;
    case 2: long  num;
    case 3: typesStruct str;
};

union myUnion3 switch(code) {
    case OPENED:      string name;
    case PROCESSED: long  num;
    case CANCELED:   typesStruct str;
};

interface ClubMed
{
    exception ClubException
    {
        string reason;
    };

    void                setClubPricesAsArray(in string club, in pricesArray prices);
    void                setClubPrices(in string club, in pricesSeq prices);
    pricesSeq           getClubPrices(in string club);
    pricesArray         getClubPricesAsArray(in string club);
    resvNo              addReservation(in string club, in resvStruct resvData);
    void                addReservationWithOut(in string club, in resvStruct
resvData, out long num);
    resvStruct          getReservation(in resvNo resv) raises ( ClubException );
    void                getReservationAsOut(in resvNo resv, out resvComplexStruct
resvData) raises ( ClubException );
    resvNo              addGetReservation(in string club, inout resvStruct
resvData);
    clubNamesSeq        getClubNames();
    clubNamesSeq4       getClubNames4();
    resvNo              addReservationComplex(in string club, inout
resvComplexStruct resvData);
    code                status(in resvNo resv) raises ( ClubException );
    boolean             cancelReservation (in resvNo resv) raises ( ClubException );
    void                setStatus(in resvNo resv, in code cod) raises (
ClubException );
    void                setCancel(in resvNo resv, in boolean cancel) raises (
ClubException );
    void                testTypes(inout typesStruct all);
    void                testOneDim(inout oneDim par);
};
```

```
        void                testTwoDim(inout twoDim par);
void                testUnion1(inout myUnion1 moon);
void                testUnion2(inout myUnion2 moon);
void                testUnion3(inout myUnion3 moon);
void                testUnion4(inout myUnion4 moon);
    };

    interface Agency
    {
        ClubMed getClub();
    };

};
};
```

---

## ClubServer.java.jacorb File

The `ClubServer.java.jacorb` file displays the Interface Repository in Java.

### **Listing C-50** `ClubServer.java.jacorb` File

---

```
// The package containing our stubs.
package bea.club;

import java.util.*;
import java.io.*;

// HelloServer will use the naming service.
import org.omg.CosNaming.*;
// All CORBA applications need these classes.
import org.omg.CORBA.*;

public class ClubServer
{
    public static void main(String args[]) {

        if (args == null || args.length == 0) {
            System.out.println("USAGE: java ClubServer (-club | -agency) [objref path]");
            return;
        }
        boolean club;
        if (args[0].equals("-club"))
```

```
        club = true;
    else if (args[0].equals("-agency"))
        club = false;
    else {
        System.out.println("USAGE: java ClubServer (-club | -agency) [objref path]");
        return;
    }

    // Create and initialize the ORB
    System.out.println("main: ORB.init(args,props)...");
    ORB orb = ORB.init(args, null);
    System.out.println("main: ORB.init(args,props) done.");

    try {
        System.out.println("try...");

        // Get a reference to the root POA
        System.out.println("main:
POAHelper.narrow(orb.resolve_initial_references(\"RootPOA\"))...");
        org.omg.PortableServer.POA poa =
org.omg.PortableServer.POAHelper.narrow(orb.resolve_initial_references("RootPOA
"));
        System.out.println("main:
POAHelper.narrow(orb.resolve_initial_references(\"RootPOA\")) done.");

        // Activate the POA
        System.out.println("main: poa.the_POAManager().activate()...");
        poa.the_POAManager().activate();
        System.out.println("main: poa.the_POAManager().activate() done.");

        // Get a ClubServant object
        org.omg.CORBA.Object o = null;
        if (club) {
            System.out.println("main: poa.servant_to_reference(new
ClubsServant())...");
            o = poa.servant_to_reference(new ClubsServant());
            System.out.println("main: poa.servant_to_reference(new ClubsServant())
done.");
        }
        else {
            System.out.println("main: poa.servant_to_reference(new
AgencyServant())...");
            o = poa.servant_to_reference(new AgencyServant());
            System.out.println("main: poa.servant_to_reference(new AgencyServant())
done.");
        }

        if (args.length == 2) {
            String oRef = orb.object_to_string(o);
```

```
        FileWriter ff = new FileWriter(args[1]);
        ff.write(oRef, 0, oRef.length());
        ff.close();
    }
else {
    // use the naming service
    // get reference to Naming Context
    System.out.println("main:
NamingContextExtHelper.narrow(orb.resolve_initial_references(\"NameService\")).
..");
    NamingContextExt nc =
NamingContextExtHelper.narrow(orb.resolve_initial_references("NameService"));
    System.out.println("main:
NamingContextExtHelper.narrow(orb.resolve_initial_references(\"NameService\")).
done.");

    // bind our object to bea.clubmed in Naming Context
    System.out.println("main: nc.rebind( nc.to_name(\"bea.clubmed\"), o)...");
    nc.rebind( nc.to_name("bea.clubmed"), o);
    System.out.println("main: nc.rebind( nc.to_name(\"bea.clubmed\"), o)
done.");
}

    System.out.println("main: orb.run()...");
    orb.run();
    System.out.println("main: orb.run() done.");

} catch(Exception e) {
    System.err.println("ERROR: " + e);
    e.printStackTrace(System.out);
}
}
}

class AgencyServant extends AgencyPOA
{
    private ClubsServant club = new ClubsServant();

    public ClubMed getClub() {
        System.out.println("method getClub() called");
        try {
            return ClubMedHelper.narrow(_default_POA().servant_to_reference(club));
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

```
class ClubsServant extends ClubMedPOA
{
    private int resNum = 999;
    private Map map = new HashMap();
    private Map mapComplex = new HashMap();
    private int resNumComplex = 999;

    public void setClubPrices(String club, pricesStruct[] prices) {
        System.out.println("method setClubPrices called");
        System.out.println("club " + club);
        for (int i = 0; i < prices.length; i++)
            System.out.println(prices[i].date + " " + prices[i].adultFare + " " +
prices[i].childFare);
    }

    public void setClubPricesAsArray(String club, pricesStruct[] prices) {
        System.out.println("method setClubPricesAsArray called");
        System.out.println("club " + club);
        System.out.println(prices[0].date + " " + prices[0].adultFare + " " +
prices[0].childFare);
        System.out.println(prices[1].date + " " + prices[1].adultFare + " " +
prices[1].childFare);
        System.out.println(prices[2].date + " " + prices[2].adultFare + " " +
prices[2].childFare);
        System.out.println(prices[3].date + " " + prices[3].adultFare + " " +
prices[3].childFare);
    }

    public pricesStruct[] getClubPrices (String club) {
        System.out.println("method getClubPrices called");
        pricesStruct[] prices = new pricesStruct[4];
        prices[0] = new pricesStruct("2002-11-23", (short)88, (short)50);
        prices[1] = new pricesStruct("2002-11-24", (short)88, (short)50);
        prices[2] = new pricesStruct("2002-11-25", (short)80, (short)45);
        prices[3] = new pricesStruct("2002-11-26", (short)80, (short)45);
        return prices;
    }

    public pricesStruct[] getClubPricesAsArray(String club) {
        System.out.println("method getClubPricesAsArray called");
        pricesStruct[] prices = new pricesStruct[4];
        prices[0] = new pricesStruct("2002-11-23", (short)88, (short)50);
        prices[1] = new pricesStruct("2002-11-24", (short)88, (short)50);
        prices[2] = new pricesStruct("2002-11-25", (short)80, (short)45);
        prices[3] = new pricesStruct("2002-11-26", (short)80, (short)45);
        return prices;
    }
}
```

```
public String[] getClubNames() {
    System.out.println("method getClubNames called");
    return new String[] { "BAMBU", "NAEBO", "RIU PALACE" };
}

public String[] getClubNames4() {
    System.out.println("method getClubNames4 called");
    return new String[] { "BAMBU", "NAEBO", "RIU PALACE" };
}

public int addReservation (String club, resvStruct resvData) {
    System.out.println("method addReservation called");
    System.out.println("club: " + club);
    System.out.println(resvData.street);
    System.out.println(resvData.city);
    System.out.println(resvData.date);
    System.out.println(resvData.fname);
    System.out.println(resvData.lname);
    System.out.println(resvData.partyAdults);
    System.out.println(resvData.partyChildren);
    System.out.println(resvData.phone);
    System.out.println(resvData.weekDate);
    System.out.println(resvData.zip);
    System.out.println(resvData.totalFare);
    map.put(new Integer(++resNum), resvData);
    System.out.println("returning " + resNum);
    return resNum;
}

public void addReservationWithOut (String club, resvStruct resvData,
org.omg.CORBA.IntHolder num) {
    System.out.println("method addReservationWithOut called");
    System.out.println("club: " + club);
    System.out.println(resvData.street);
    System.out.println(resvData.city);
    System.out.println(resvData.date);
    System.out.println(resvData.fname);
    System.out.println(resvData.lname);
    System.out.println(resvData.partyAdults);
    System.out.println(resvData.partyChildren);
    System.out.println(resvData.phone);
    System.out.println(resvData.weekDate);
    System.out.println(resvData.zip);
    System.out.println(resvData.totalFare);
    map.put(new Integer(++resNum), resvData);
    num.value = resNum;
}

public resvStruct getReservation (int resv) {
```

```
        System.out.println("method getReservation called");
        resvStruct resvData = (resvStruct)map.get(new Integer(resv));
        System.out.println("Returning: " + resvData);
        return resvData;
    }

    public int addGetReservation (String club, resvStructHolder resvData) {
        System.out.println("method addGetReservation called");
        resvStruct data = resvData.value;
        System.out.println(club);
        System.out.println(data.street);
        System.out.println(data.city);
        System.out.println(data.date);
        System.out.println(data.fname);
        System.out.println(data.lname);
        System.out.println(data.partyAdults);
        System.out.println(data.partyChildren);
        System.out.println(data.phone);
        System.out.println(data.weekDate);
        System.out.println(data.zip);
        System.out.println(data.totalFare);
        System.out.println(data.resv);
        map.put(new Integer(++resNum), data);
        return resNum;
    }

    public boolean cancelReservation (int resv) {
        System.out.println("method cancelReservation called");
        System.out.println("returning false");
        return false;
    }

    public int addReservationComplex(java.lang.String club, resvComplexStructHolder
resvData) {
        System.out.println("method addReservationComplex called");
        mapComplex.put(new Integer(++resNumComplex), resvData.value);
        System.out.println("Returning: " + resNumComplex);
        return resNumComplex;
    }

    public bea.club.code status (int resv) {
        System.out.println("method status called");
        System.out.println("reserv: " + resv + " returning: " +
        bea.club.code.PROCESSED);
        return bea.club.code.PROCESSED;
    }

    public void setStatus(int resv, bea.club.code cod) {
```

```
        System.out.println("method setStatus called");
        System.out.println("reserv: " + resv + " cod: " + cod);
    }

    public void setCancel(int resv, boolean cancel) {
        System.out.println("method setCancel called");
        System.out.println("reserv: " + resv + " cancel: " + cancel);
    }

    public void testTypes(typesStructHolder all) {
        System.out.println("method testTypes called");
        typesStruct t = all.value;
        System.out.println(t.t1);
        System.out.println(t.t2);
        System.out.println(t.t3);
        System.out.println(t.t4);
        System.out.println(t.t5);
        System.out.println(t.t6);
        System.out.println(t.t7);
        System.out.println(t.t8);
        System.out.println(t.t9);
        System.out.println(t.t10);
        System.out.println(t.t11);
        System.out.println(t.t14);
        System.out.println(t.t15);
    }

    public void testOneDim(oneDimHolder par) {
        System.out.println("method testOneDim called");
        int[] one = par.value;
        for (int i = 1; i <= one.length; i++) {
            System.out.println("item " + i + " " + one[i-1]);
        }
    }

    public void testTwoDim(twoDimHolder par) {
        System.out.println("method testTwoDim called");
        String[][] two = par.value;
        for (int i = 1; i <= two.length; i++) {
            System.out.println("row " + i);
            for (int j = 1; j <= two[i-1].length; j++)
                System.out.println("item " + j + " " + two[i-1][j-1]);
        }
    }

    public void testUnion1(bea.club.myUnion1Holder moon) {
        System.out.println("method testUnion1 called");
    }
}
```



```
public void testUnion2(bea.club.myUnion2Holder moon) {  
    System.out.println("method testUnion2 called");  
}  
  
public void testUnion3(bea.club.myUnion3Holder moon) {  
    System.out.println("method testUnion3 called");  
}  
  
public void testUnion4(bea.club.myUnion4Holder moon) {  
    System.out.println("method testUnion4 called");  
}  
  
}
```

---

